Neural Observers for Non-Linear Systems

by

Sayyid Hasan Riyaz

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

November, 1997

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

NEURAL OBSERVERS FOR NON-LINEAR SYSTEMS

BY

SAYYID HASAN RIYAZ

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS ENGINEERING

NOVEMBER 1997

KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS

DHAHRAN, SAUDI ARABIA

This thesis, written by

Sayyid Hasan Riyaz

under the direction of his Thesis Advisor, and approved by his Thesis committee, has been presented to and accepted by the Dean, College of Graduate Studies, in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

Thesis Committee:

Md. Shafgir Ahmed

Chairman (Dr. M. Shafgir Ahmed)

Member (Dr. Fouad M. Al-Sunni)

Member (Dr. H.E. Emara-Shabaik)

Dr. A. A. Andijani
Department Chairman

Dr. A M. Al-Shehri
Dean College of Graduate Studies

Date: 3-12-97

كلية الدراسات العليا
COLLEGE OF GRADUATE STUDIES

NEURAL OBSERVERS FOR NON-LINEAR SYSTEMS

Sayyid Hasan Riyaz

Systems Engineering

November 1997

Dedicated to

My Parents

Acknowledgments

First of all I wish to thank my thesis advisor Dr. M. Shahgir Ahmed for his countless hours of guidance inspite of his extremely busy schedule. Thanks are also due to my thesis committee members for their advice and constructive criticism.

I am also indebted to the department chairman, Dr. Abdul-Basit Andijani and other faculty members for their support.

Lastly, but not the least, thanks are due to my family members and friends for their encouragement and support through out my stay in KFUPM. Their support and company made it possible for me to work through long hours for the completion of this work.

Contents

List of Figures

List of Tables

Abstract

Name: Sayyid Hasan Riyaz

Title: Neural Observers for Non-Linear Systems

Major Field: Systems Engineering

Date of Degree: November 1997

Design of dynamic neural observer for nonlinear plant is considered. Six different schemes have been proposed. Simulation studies have been conducted to establish their usefulness and evaluate their performance. Simulation study also established the superiority of the proposed schemes over the static neural observer and the extended Kalman filter.

Master of Science Degree
King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
November, 1997

خلاصة الرسالة

الإسم: سيد حسن رياض

العنوان: تصميم ملاحظ لتقدير حالة الأنظمة الغير خطية باستخدام الشبكات العصبية

التخصص: هندسة نظم

تاريخ الشهادة: نوفمبر 1997

نقترح ستة طرق لتصميم ملاحظ ديناميكي لتقدير حالة الأنظمة الغير خطية. لقد قمنا بدراسةٍ مقارنةٍ مع بعض الطرق الأخرى مثل الملاحظ الغير ديناميكي و مصفي كالمن للنظم الغير خطية ولقد أظهرت الدراسات المعتمدة على المحاكاة والتي قمنا بها تحسناً ملحوظاً في أداء الملاحظات المقترحة عند مقارنتها بالطرق الأخرى لتقدير حالة النظام.

ماجستير في العلوم

جامعة الملك فهد للبترول والمعادن

الظهران- المملكة العربية السعودية

نوفمبر 1997

Chapter 1

Introduction

The problem of identification of a system is dual to the problem of controlling a system, as no system can be controlled if it is not identified, either before or during the application of control. Apart from identification and control design another ingredient in control application is the state estimation. Many control schemes rely on the availability of the state estimates. The state of a process is defined as a variable which together with the subsequent input to the system, completely determines the subsequent behavior, i.e. the knowledge about the state at a certain time and the subsequent input (together with the knowledge of process structure and parameters) are sufficient to predict its future behavior [1]. In addition to its application in control, state estimation is also useful for process monitoring and fault diagnosis.

The mathematical systems theory for linear systems is a well established field and

is based on linear algebra, complex variable theory and the theory of ordinary linear differential equations, dealing with the analysis and synthesis of dynamic systems. Since design techniques for dynamical systems are closely related to their stability properties and since necessary and sufficient conditions for the stability of linear time-invariant systems have been generated over the past decades, well-known design methods have been established for such systems [2].

However, most of the practical systems are nonlinear and it is a general practice of engineers to use linear models for the analysis and design of physical systems. This simplifies the analysis and design of the system, but the controller based on this model may lead to performance deterioration and may also make the control system unstable. To improve the performance, the model has to be refined and some nonlinearities may appear in the model, which emphasize the importance and difficulty of the analysis and design of nonlinear systems.

The study of nonlinear systems modelling and control, of which state estimation is a part, is an important issue which is still in the developing stage. Most of the techniques developed, first linearize the system and then apply the techniques for the linear systems.

The most popular method for state estimation in non-linear systems is the Extended Kalman Filter (EKF). The idea behind the extended Kalman filter (EKF) is to adapt the linear Kalman filter to non-linear systems. This is achieved by lin-

earization of the system around the current state estimates and application of the Kalman filter to the resulting (time varying) linear system. The resulting algorithm is computationally quite demanding [3].

Another approach for state estimation of nonlinear systems with discrete measurements is through the use of Newton's algorithm. Discrete Newton algorithm if properly interpreted yields an asymptotic observer for a large class of discrete time systems. The method relies on asymptotically inverting the state-to-measurement map, which is formed by relating the system's states at a given time to a predetermined number of consecutive measurements. This algorithm is computationally very complex and expensive due to repeated Jacobian evaluations. The continuous Newton method yields a global observer, but computations are even more complex than the discrete one. Approximations of Jacobians can also be used by employing the Broyden's method, but occasional recalculation of Jacobians remains necessary [4].

During the recent years a lot of research work is done on the artificial neural networks and its application in the field of nonlinear systems identification and control. Various researchers have suggested a promising role for the neural networks in the coming future. The ability of the neural networks to learn and approximate nonlinear functions is perhaps the most significant reason of its wide application in the field of nonlinear systems.

In this thesis, design of dynamic neural observer is considered. Six different

schemes are proposed. Back-propagation algorithm in conjunction with the BPD algebra are used to train the network. Adaptive learning rate is applied to the neural network to facilitate fast learning. The bulk of the computational effort of the proposed nonlinear observers is during its training (design) stage. The computational effort during the implementation stage is trivial that makes it suitable for control and diagnosis applications.

Chapter 2

Literature Review

In this chapter a literature survey of different techniques used in the design of nonlinear observers is given. Applications of neural networks for the design of nonlinear observers are looked into and a detailed literature survey is included.

State estimation of linear systems has been thoroughly studied and documented. Corresponding studies for nonlinear systems are still developing. In majority of these studies attempts have been made to linearize or quasi-linearize the nonlinear model in order to extend the linear approaches.

The standard approach to state estimation in nonlinear systems is the Extended Kalman Filter (EKF) which has been reported in details in Jazwinski [5]. It is the most popular approach and is a form of Kalman filter 'extended' to the nonlinear dynamical systems. It is also developed in a predictor-corrector format while the

extensions include linearization at different stages of the algorithm using Taylor series linearization with the assumption that the higher order terms can be neglected. It also assumes that the covariances of the noise processes are known. The performance of EKF deteriorates or may even diverge in the case of erroneous models [6]. EKF must be viewed as an adhoc filter as no convergence result is known to exist [7]. The convergence analysis of the EKF are given by Boutayeb et al. [8] and it is concluded that with a proper choice of the covariance matrices, the convergence of the EKF can be improved significantly even for a very bad initialization. Iterative correction term can also be used to improve the performance of the EKF.

Dhingra et al. [9] presented a Jump Matrix Technique (JMT) for the state estimation problem in nonlinear systems. The method is based on the same footings as the basic Kalman state estimator and approximates the nonlinear system as a memoryless nonlinearity embedded in a dynamic linear system using fictitious samplers and clamps. Quasi-linearization is used in place of Taylor series linearization of the Extended Kalman filter. During a sample interval, the nonlinear dynamics of the system are held constant until the end of the sample interval at which time they 'jump' to an updated value. Simulation studies indicated that JMT technique can be used effectively in cases where the EKF fail to converge.

Ahmed [6] derived an innovation model for a nonlinear stochastic system described by a state variable representation. A robust state and parameter estimation method

is proposed through minimization of the innovation variance (MIV). Performance of the algorithm is compared with EKF and JMT. It has been shown that in the presence of nonstationary state and measurement noise, when the statistics of the state and measurement noise are not known properly, and/or, when the system model is imprecise the proposed MIV algorithm outperforms the EKF-type algorithms. MIV always attempts to minimize the prediction error variance by adjusting the innovation gain matrix and therefore has a superior model mismatch tolerance. The author underlines the major weakness of the MIV algorithm to be the lack of a rigorous stability proof.

Moraal and Grizzle [4] proposed an observer design for nonlinear systems with discrete time measurements by using Newton's method. The basic idea is to solve a set of nonlinear inversion problems, formed by relating the system states at a given time to a predetermined number of consecutive measurements, to get an asymptotic observer. They established that if discrete Newton's method is properly interpreted, it will yield an asymptotic observer for a large class of discrete time systems. If continuous Newton algorithm is applied then a global observer can be obtained. These schemes are computationally very complex and expensive. Although the authors have used Broyden's scheme for the approximation of Jacobians to make it computationally efficient, it has also been mentioned that occasional evaluation of Jacobians will remain necessary if not in each step. The Newton's algorithm is also used for the

observer design of poorly observable and detectable nonlinear systems [10].

Lie algebraic or global linearization technique is another technique used in the observer design of nonlinear systems. Bestle and Zeitz [11], who first introduced this concept, transformed the system in canonical form such that the transformed error dynamics are linear. Then the observer design can be performed using any linear design technique. Krener and Respondek [12] modified this algorithm to accommodate nonlinear control inputs. The main disadvantage of this algorithm is that it is not always possible to get a nonlinear transformation to convert the system to a canonical form. A precise knowledge of the system is also required.

Ramnath and Paynter [13] used the idea of a scaling transformation to reduce the nonlinear problem to an obviously solvable form. For example, the nonlinear Riccatti equation: $\dot{z} + z^2 = t$ can be made equivalent to the linear equation: $\ddot{y} - ty = 0$ by using the change of variables $z = \dot{y}/y$. The main disadvantage of this method is that it is generally very difficult to get an appropriate transformation.

Pseudo-linearization techniques using successive transformations to convert the nonlinear system to an observerable canonical form are also used for the state estimation of nonlinear systems as given by Nicosia et al. [14] and Reboulet et al. [15]. Once the canonical form is achieved the observer design becomes very simple and is similar to Lie algebraic technique. With this technique a certain class of problems for which Lie algebraic technique fails, a canonical form can still be obtained. The

disadvantages are the same as that of the previous technique in addition that only local properties along a set of operating points are guaranteed.

Another approach for nonlinear state estimation is the extended linearization technique. Bauman [16] showed that the observer error equation, when linearized about a neighborhood of a set of operating points of the system possess locally invariant eigen values. The eigen values are constant in a sufficiently small neighborhood of the operating points. However, in the presence of disturbance or modelling errors the performance of extended linearization technique deteriorates [17]. The method also requires an exact knowledge of the system dynamics.

Thau's method, introduced by Thau [18], does not give a systematic observer design method, rather it is a verification method which gives certain conditions for the convergence of estimates for a given system. If the conditions are not satisfied then the technique guarantees stability only. Thau's observer incorporates the nonlinearities of the system into the observer design although it does not handle modelling errors. Certain knowledge of the nonlinearities of the system are also essential.

Crandall [19] deals with nonlinear systems through statistical linearization approach by replacing the nonlinear system equation with an equivalent linear equation involving a parameter which is selected to minimize the estimation error. The main step is the evaluation of statistical expectations of certain functions of the nonlinear response. This approach only works for small nonlinearities. Various other nonlinear

observer design techniques and their comparative study can also be found in Misawa and Hedrik [17] and Walcott et al. [20].

The Extended Luenberger Observers presented by Zeitz [21] are analogous to the Luenberger observer for linear systems. These are developed on the same principle as the extended Kalman filter. local linearization of the nonlinear system around the reconstructed states. Kazantzis and Kravaris [22] transformed the system into first order partial differential equations and then these equations are solved by applying Lyapunov's auxiliary theorem. A general set of necessary and sufficient conditions for solvability are derived. Other Luenberger type observers are given by Ciccarella et al. [23] and Gauthier et al. [24]. A Luenberger type observer for discrete time nonlinear system is also given by Ciccarella et al. [25].

Sliding Mode Observers (SMO) are nonlinear observers based on the theory of Variable Structure Systems (VSS). These are basically Luenberger observers with an additional switching term for robustness against modelling errors and uncertainties. SMO for continuous time system is given by Drakunov [26]. Its extensions to more general nonlinear systems are given by Slotine et al. [27] and [28]. A comparative study can be found in [17] and [20]. A tutorial on sliding mode observers is presented by Drakunav and Utkin [29] giving systematic procedures for the design of sliding mode observers for linear and nonlinear systems. Aitkin and Schwartz [30] claim to be the first ones in developing discrete time sliding mode observer. The main advan-

tage of the sliding mode observers is their robustness against unmodelled dynamics provided that the sliding conditions are maintained.

Recently some results are also reported in the application of Genetic Algorithm (GA) for state estimation in nonlinear systems. Porter and Passino [31] claim to be the first ones to use genetic algorithm for online state estimation. Genetic algorithm is based on the principles of evolution, natural selection and genetics to offer a method for parallel search of complex spaces. Receiving the process input and output information, the GA manipulates the observer gains to reduce the state estimation error to zero. The approach is in the primitive stage and although some promising results are given, no mathematical analysis is provided yet.

Artificial neural networks is a rapidly expanding field which is evident by the explosive growth of papers, journals, conferences and conference sessions devoted to it. Control systems is one of the fields where the artificial neural networks has found tremendous applications. In late 80's and early 90's neural networks were used in nonlinear systems identification and control.

Page et al. [32] gave a general introduction to the neural networks with theoretical discussion, implementation of neural networks in process modelling and estimation, and applications of various neural network control strategies. The important characteristics of neural networks which are of interest to control engineers and their application are described as:

- Modelling: Because of their ability to learn using data records for the particular system of interest, the major problem of developing a realistic system model is obviated.

- Non-linear systems: The networks possess the ability to 'learn' non-linear relationships with limited prior knowledge about the process structure. This is possibly the area in which they show the greatest promise.

- Multivariable systems: Neural networks, by their very nature, have many inputs and many outputs and so can be readily applied to multivariable systems.

- Parallel structure: The structure of neural networks is highly parallel in nature. This is likely to give rise to three benefits: very fast parallel processing, fault tolerance and robustness.

Cybenko [33] and Hornik et al. [34] have described neural networks as universal function approximators. It has been shown that a continuous function can be approximated with arbitrary accuracy by a multilayer feedforward neural network.

An extensive survey for the application of neural networks in control systems is done by Hunt et al. [35]. The main focus is on the impact of neural networks in the realm of modelling, identification and control of nonlinear systems. The basic ideas and techniques with applications of a variety of neural networks architectures in control and directions for future research are given.

Chen et al. [36] used backpropagation technique to train a neural network model for system identification and has shown that the backpropagation algorithm is a special case of prediction error algorithm.

Werbos [37] proposed a modified back propagation algorithm known as back propagation through time (BPTT) to compute the gradients of the neural networks involving dynamic systems. Application of this new algorithm to the areas of pattern recognition, system identification and control etc. are discussed.

A detailed discussion is given by Narendra and Parthasarathy [2] on the effectiveness of neural networks for identification and control of nonlinear dynamical systems. Various methods for utilizing multilayer and recurrent networks for identification and control have also been given therein. Only minimum phase plants of relative degree unity have been considered and the structure of the plant is required to be known. The dynamic backpropagation (DB) technique is introduced for the training of the neural network. A detailed description of this technique can be found in [38].

Willis et al. [39] discussed the application of neural networks as an inferential estimator and a nonlinear predictive controller for fast inference of a difficult to measure process output from other easily measured variables. They used the estimates produced by the neural estimator as feedback signals for control to improve the process regulation.

Levin and Narendra [40] discussed the identification of nonlinear systems using

neural networks. Design of practically viable controllers using neural networks, based on the nonlinear systems control theory are discussed in [41]. The assumptions have been that the states of the system are accessible.

In a subsequent article [42], the observability, identification and control of nonlinear systems using neural networks are discussed. It is assumed that the model of the system is known. A static neural observer is designed for state estimation of the nonlinear system. The approach is based on the realization that if the linearization of the system around an equilibrium point is controllable and observable, then locally the nonlinear system will exhibit behavior that is topologically equivalent to that of the linear system. The static neural observer design will be further discussed in chapter 5.

The literature survey included different design techniques of nonlinear observers and shows that it is still a growing field. Neural networks due to their capability of approximating nonlinear systems, made them significantly important in this field with wide scope of applications. Design of static neural observers has already appeared in the literature. Considering the effectiveness of neural networks, we are exploring the design of dynamic neural observers for nonlinear systems. Recently developed techniques of block partial derivatives by Ahmed [43, 44] will be employed to simplify the gradient calculation in the dynamic neural observers.

Chapter 3

Artificial Neural Networks

This chapter gives a brief introduction to the artificial neural networks. The basic concepts of feed-forward neural networks along with the learning algorithms are described. Back-propagation, adaptive learning and the technique of block partial derivatives are included.

3.1 Introduction

It has always been a human desire to mimic the systems of human body. Fascinated by the human learning process, Artificial Neural Networks (ANN) were inspired by the pattern of human nervous system and the operation of human brain, and were developed in a manner that resembles the functions of biological neurons. The development of artificial neural networks began in early 1940s, with the earliest networks

being developed by McCulloch and Pitts in 1943. Several other models were developed during that era. In early 1960s Frank Rosenblatt introduced and developed a large class of neural networks called perceptrons. Due to the restrictions of the single layer perceptrons and also the lack of a general structure for training the neural network there was little advancement in 1970s. Although Werbos in 1974 developed the OPD (ordered partial derivatives) concept and back-propagation algorithm to propagate the output error back to the hidden layers but the real enthusiasm in this field was developed in 1980s when Rumelhart and others refined and publicized the algorithm in 1986 [45]. With the advent of the modern computers and advanced computational facilities there has been an exponential rise in the research contributions to this field [32, 46]. Recently the research work has been formalized as standard texts and quite a few books can be found on this subject [47, 48].

3.2 Architecture

Artificial neural networks are information processing systems that are inspired by the biological neural networks. The architecture mainly consists of very simple elements called neurons connected together to form a network. The most commonly used neuron is shown in Fig 3.1 and is based on the model proposed by McCulloch-Pitts in 1943.

The inputs to a neuron, $x_1, \ldots, x_n$, are applied through separate links and all

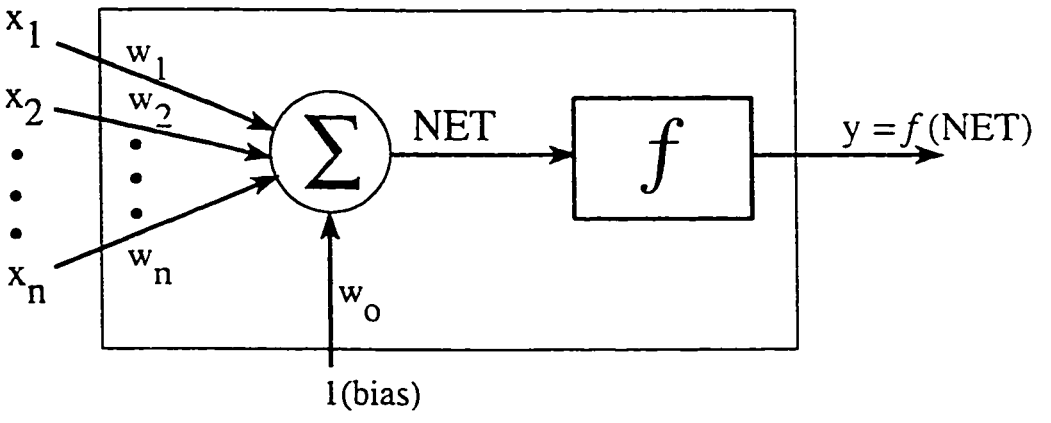


Figure 3.1: An artificial neuron

links are weighted by their corresponding weights $w_1, \ldots, w_n$. A bias in the node is characterized as an additional constant input of '1' weighted by the value w_0. After multiplying by their associated weights, the summation block adds up all the inputs and gives the output NET which can be written as

$$NET = \sum_{i=1}^{N} w_i x_i + w_o \tag{3.1}$$

3.3 Activation Functions

Usually the output NET of the neuron obtained by summing the weighted inputs is passed through a non-linear activation function, $f(.)$, to get a final output y:

$$y = f(NET) = f\left(\sum_{i=1}^{N} w_i x_i + w_o\right) \tag{3.2}$$

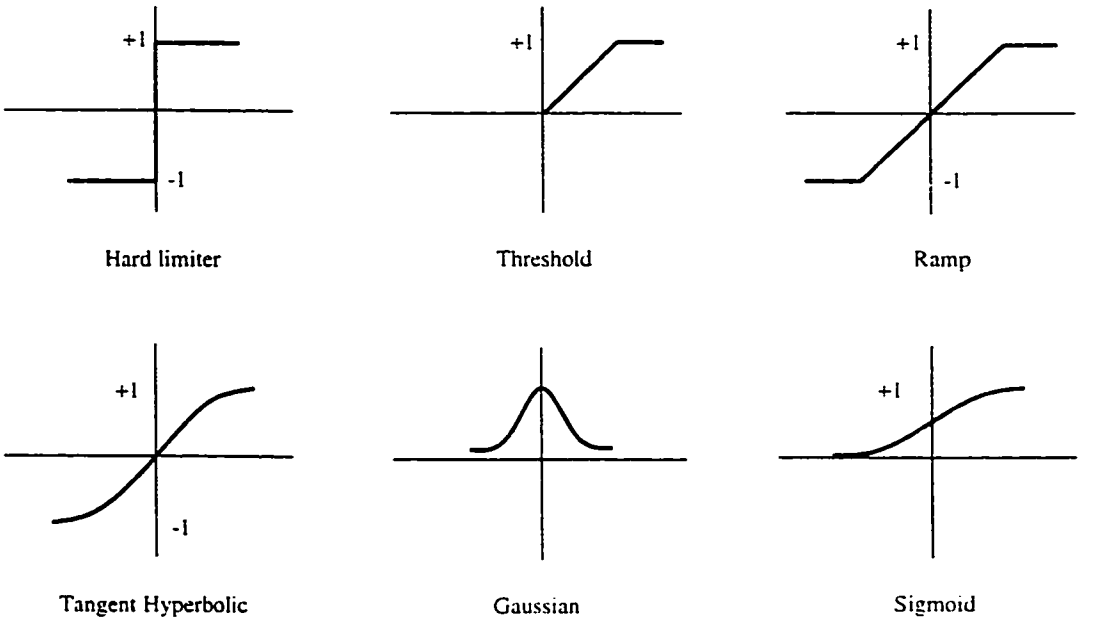


Figure 3.2: Various activation functions

Various types of non-linear functions can be used to get the final output. Some of the common activation functions are shown in Fig 3.2.

Cybenko [33] has shown that a continuous function can be approximated with arbitrary accuracy by a superposition of sigmoidal functions. The activation function typically used in system identification, estimation and control is the hyperbolic tangent function which is obtained by adding a bias and rescaling the sigmoidal function. The mathematical expression is given as

$$y = tanh(x)$$

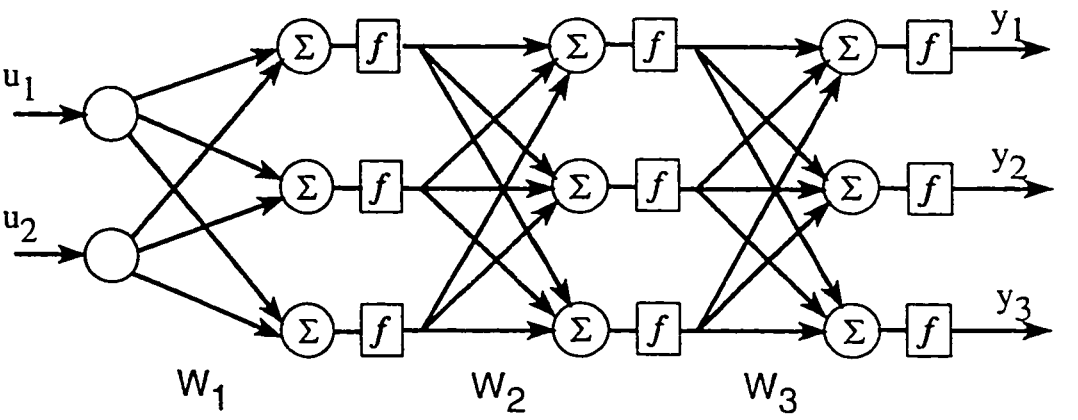


Figure 3.3: Multi-layer Feed-forward NN (MFNN)

$$\text{or} \qquad y = \frac{1 - e^{-NET}}{1 + e^{-NET}} \tag{3.3}$$

The hyperbolic tangent function is an S-shaped squashing function symmetrical about the origin as shown in the Fig 3.2. The value of y becomes zero when NET is zero. It also compresses the range of the NET, so that the output y never exceeds the range $[+1, -1]$ regardless of the value of NET.

3.4 Multi-layer Feed-forward Neural Networks

The arrangement of the neurons in such a network consists of an input layer, a number of hidden layers and an output layer as shown in the Fig 3.3. While neurons, as a basic element are not that powerful but once connected together in such a configuration their processing capabilities are enhanced considerably.

This is the most popular neural network architecture. The input layer is essentially a direct link to the inputs of the first hidden layer while the hidden layers are made up of the neuron cells as shown in the Fig 3.1. Each neuron in a layer is directly connected to all the neurons in the next layer. Neurons in the same layer are not connected together. Data flow in this kind of network is unidirectional i.e. from input to the output in forward direction. These networks are also known as multi-layer perceptrons. For most of the applications, a network of one to three hidden layers are quite sufficient.

The output layer in our application is also like hidden layer except that the squashing nonlinear function is not included here to remove the [+1,-1] bound. Tangent hyperbolic functions for the neurons in the hidden layers are the most appropriate choice for the control applications while other functions may also be used.

3.5 Training of Feed-forward Neural Networks

The popularity of the neural networks is based on its learning capability. In a typical application, given a set of inputs the neural network is supposed to train itself using supervised learning techniques so that it can reproduce the desired outputs. The recent boom in the research activities of neural networks is based on the back-propagation algorithm presented by Rumelhart [45]. Back-propagation technique is used for the training of the neural network. Due to its simplicity, this technique

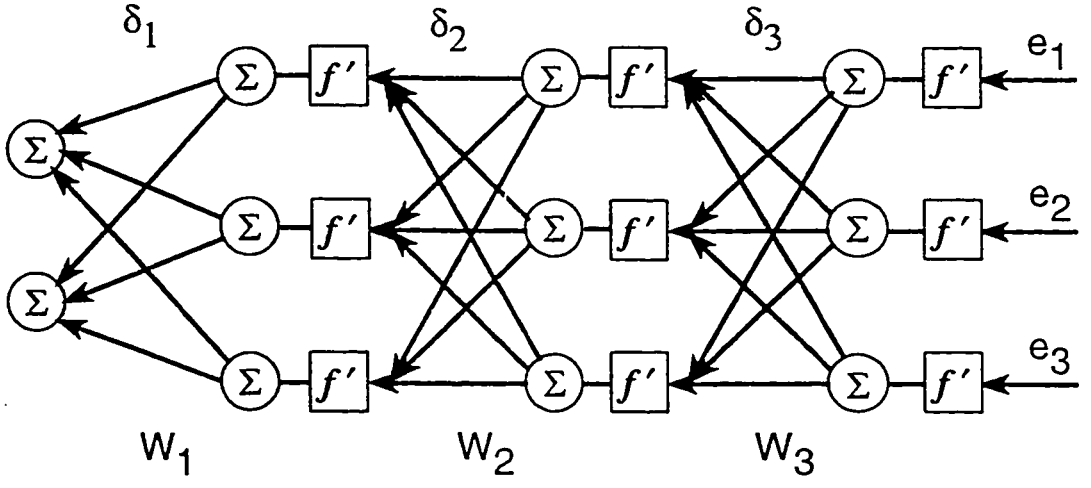


Figure 3.4: Error back-propagation

played a vital role in the renewed interest and application of neural networks in a wide range of problems.

The back-propagation technique is based on the gradient descent method to minimize the error between the actual neural network output and the desired output. Back propagation is referred to the systematic procedure of computing the gradient with respect to the neural network weights. The network is first initialized with a random set of weights. The input set of data is applied to the neural network and the output of the network is calculated through forward pass. The output is compared with the desired output and the error so generated is back-propagated from the output to the input of the network using the delta rule as shown in Fig 3.4. The network weights are adjusted by the gradients so as to minimize a cost function, typically the sum of

the squared errors. The weight adjustment process is repeated iteratively until the error is reduced to a desired level or a maximum number of iterations is reached, in which case the network configuration is modified by increasing the number of hidden layers or the number of neurons in the hidden layers and the network is trained again.

The back-propagation is a first order gradient scheme to minimize the error between the actual and the desired output. Let y and y_d be the actual and the desired scalar outputs of the network respectively and let e_k be the difference between y and y_d termed as the error.

Back-propagation gives a systematic procedure to compute the gradient $\partial J/\partial W$ where,

$$\begin{aligned} J &= \frac{1}{2}\|e_k\|^2 \\ &= \frac{1}{2}\|y_d - y\|^2 \end{aligned} \tag{3.4}$$

The algorithm minimizes this cost function J by employing the well known delta rule by updating the weights. Delta rule is a first order gradient descent algorithm moving in the negative gradient direction. By applying the delta rule to the network we can get the weight update equation as

$$W_{k+1} = W_k - \eta \left[\frac{\partial^- J}{\partial W}|_{W=W_k}\right]^T \tag{3.5}$$

where η known as the "learning rate", is a small positive quantity chosen arbitrarily

and $\partial^+ J/\partial W$ denotes the ordered partial derivative [37]. Defining Γ_k as

$$\Gamma_k = \frac{\partial^+ y}{\partial W}|_{W=W_k} \tag{3.6}$$

and using the equation for cost function Eq 3.5 (also see Fig 3.1), one gets

$$\begin{aligned} \frac{\partial^- J}{\partial W} &= -\Gamma_k e_k \\ &= -\frac{\partial^+ y}{\partial W} e_k \\ &= -\frac{\partial^+ y}{\partial NET}\frac{\partial NET}{\partial W} e_k \end{aligned} \tag{3.7}$$

From where we get the weight update equation as

$$W_{k+1} = W_k + \eta_k \Gamma_k e_k \tag{3.8}$$

The activation function of the neurons is taken as the tangent hyperbolic function given by the Eq. 3.3. The derivative of this function can be easily calculated and is given by,

$$\frac{\partial^- y}{\partial NET} = \frac{1}{2}(1+y)(1-y) \tag{3.9}$$

Also, from Eq. 3.1 we have,

$$\frac{\partial^+ NET}{\partial W} = x \tag{3.10}$$

After substituting we get the weight update equation for the *output layer* as,

$$W_{k+1} = W_k + \eta\frac{1}{2}(1+y_k)(1-y_k).x_k.e_k \tag{3.11}$$

where,

W_{k+1} = the updated weight in the output layer

W_k = the old weight in the output layer

y_k = the output produced by the neuron

x_k= the corresponding input to the neuron

Back-propagation algorithm can be applied to networks with any number of layers. and the above equation can be extended to update weights in other layers too [45].

3.6 Backpropagation

In this section an overview of the composite training process of a neural network is presented. The main purpose of the training is to adjust the weights of the neural network so that it can reproduce the desired output. Each input vector to the network is paired by a corresponding target vector. The procedure is started by assigning random weights to the network. The input vector is presented to the network and the output of the network is calculated through a forward pass. The output is compared to the target vector corresponding to the input vector. The error between the two is calculated and then the weight is adjusted to reduce this error. This is an iterative process until the desired minimum error is achieved. The whole process can be split into two phases. forward pass. where the output is calculated and the reverse pass. where the weight update is achieved.

3.6.1 Forward Pass

Let the weight matrix between two consecutive layers i, j be W_{ij}, and let the input vector of layer i be X_i then the output of layer i, Y_i,will be given as

$$Y_i = f(X_i^T W_{ij}) \tag{3.12}$$

where $f(.)$ is a nonlinear activation function. This output Y_i will be the input X_j to the next layer j. This process is repeated for each layer until the output of the last layer is obtained.

3.6.2 Reverse Pass

In this phase the weights of the neurons are adjusted to minimize the error. From the forward pass the output of the network is obtained for a given set of input. Consider the kth layer as the output layer and the corresponding being Y_k. In this pass, first the output Y_k is subtracted by the desired output Y_d to get the error e_k as

$$e_k = Y_d - Y_k \tag{3.13}$$

This error e_k is multiplied by the derivative of the squashing function. If the nonlinear squashing function f is hyperbolic tangent as given in Eq 3.3, then its derivative is given as

$$f'(.) = \frac{1}{2}(1 - f(.))(1 + f(.)) = \frac{1}{2}(1 - f(.)^2) \tag{3.14}$$

Defining delta (∂_k) for the kth layer as

$$\partial_k = \frac{\partial J}{NET_k} = \frac{\partial^+ Y_k}{\partial NET_k} e_k \tag{3.15}$$

we get

$$\partial_k = \frac{1}{2}(1 - Y_k)(1 + Y_k)e_k \tag{3.16}$$

which is then multiplied by the layers input which is actually the output of the preceeding layer. So we get

$$\frac{\partial J}{\partial W_{jk}} = \partial_k Y_j \tag{3.17}$$

ΔW, i.e. the change in weight is obtained as

$$\Delta W_{jk} = \eta \frac{\partial J}{\partial W_{jk}} = \eta \partial_k Y_j \tag{3.18}$$

Hence the updated value of the weight is given by

$$W_{jk}^{new} = W_{jk}^{old} + \Delta W_{jk} \tag{3.19}$$

The updated weights for the hidden layers can be obtained by back-propagating the error from the output layer to the hidden layer to obtain the appropriate partial derivative via the chain rule. For each hidden layer the ∂ is calculated by multiplying the ∂ from the next layer by the weights between the two layers. Mathematically we can write,

$$\partial_j = \frac{1}{2}(1 - Y_j)(1 + Y_j)(\Sigma \partial_k W_{jk}) \tag{3.20}$$

So the weights in the hidden layer, i, can be adjusted by

$$W_{ij}^{new} = W_{ij}^{old} + \eta\frac{1}{2}(1+Y_j)(1-Y_j)(\Sigma\delta_k W_{jk})Y_i \tag{3.21}$$

where,

W_{jk}^{new} = the updated weight in the output layer k

W_{jk}^{old} = the old weight in the output layer k

δ_k = the value of δ in the output layer k

Y_k = the output of layer k.

X_k = the input to layer k.

The learning rate "η" is usually chosen by trial and error. The above learning is known as the pattern learning, where the weights are updated after presentation of each pattern. Another learning known as "batch learning" constitutes of updating the weights after a complete presentation of all patterns. Batch learning is also iterative and repeats the presentation of all patterns until the network learns adequately [45].

3.7 Block Partial Derivative (BPD)

Initiated by Back Propagation (BP) technique published by Rumelhart [45] different techniques are proposed in the literature to calculate the derivatives for the learning process of the recurrent neural networks. A neural network is termed as "recurrent" when dynamics are added to it through any form of feedback. A feedback may be

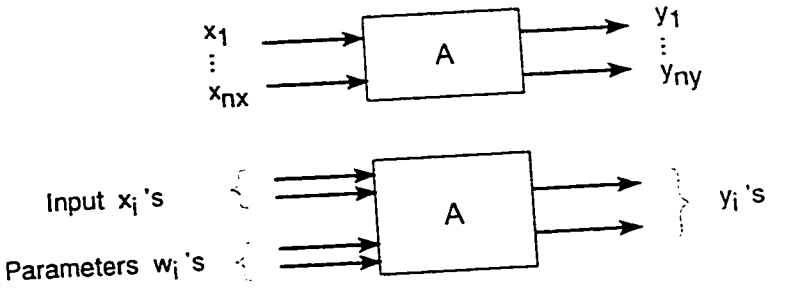


Figure 3.5: Block Partial Derivative (BPD)

added to a neural network either internally or externally. The techniques found in the literature are Back Propagation Through Time (BPTT) developed by Werbos [37], Dynamic Backpropagation (DB) developed by Narendra and Parthasarathy [2, 38] and Block Partial Derivatives (BPD) proposed by Ahmed [43, 44]. All of these algorithms are the extensions of BP algorithm to the dynamic systems.

Block Partial Derivative (BPD) is a generalization of the concept of DB and the ordered partial derivatives (OPD). Ahmed [43] introduced this concept and gave the algebra for the partial derivative computation in a system when different blocks are connected in various configurations. The BPDs are defined to be similar to the ordinary partial derivatives except that there is a block, which may be imaginary, characterizing the explicit dependence of a variable on other variables [44]. These other variables are analogous to the independent variables in the ordinary partial derivatives. Considering a certain block A with input $x \in R^{nx}$ and output $y \in R^{ny}$, the

BPDs are defined as

$$\frac{\partial^A y_i}{\partial x_j} = lim_{\Delta x_j \to 0} \frac{\Delta y_i}{\Delta x_j} \qquad \Delta x_k = 0 \quad for \quad k \neq j \qquad (3.22)$$

The BPD of the output of a block with respect to a particular input is computed by considering all other inputs to the block to be constant. However other signals within the block are allowed to vary owing to the change in the particular input. The varying parameters inside a block can be accommodated by considering them as auxiliary inputs to the block as shown in Fig 3.5, which facilitates the computation of BPDs with respect to these parameters.

The BPDs may assume dynamic operators if the inputs and outputs of the block are functions of time and the block is dynamic. The procedure for calculating BPDs for different types of block configurations is given in [43].

A simpler approach for the computation of BPDs is also given by Ahmed [44]. Unlike DB and BPTT which employ chain rule for derivative computation, this approach is based on the Mason's rule [49] of signal flow graph reduction. In this approach a linearized network is obtained by replacing each block with its linearized model. The linearized network is then reduced by using the signal flow graph reduction technique (Mason's rule) which avoids the cumbersome task of step-by-step block diagram reduction.

As for an example consider a nonlinear dynamic system shown in the Fig 3.6 and its linearized network is given in the Fig 3.7 After the application of the Mason's the

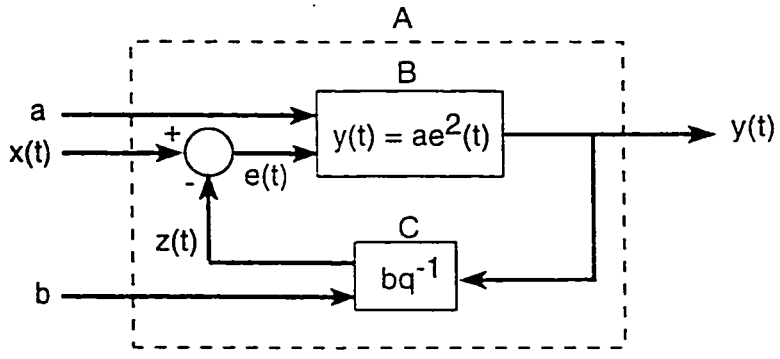


Figure 3.6: A nonlinear dynamic system

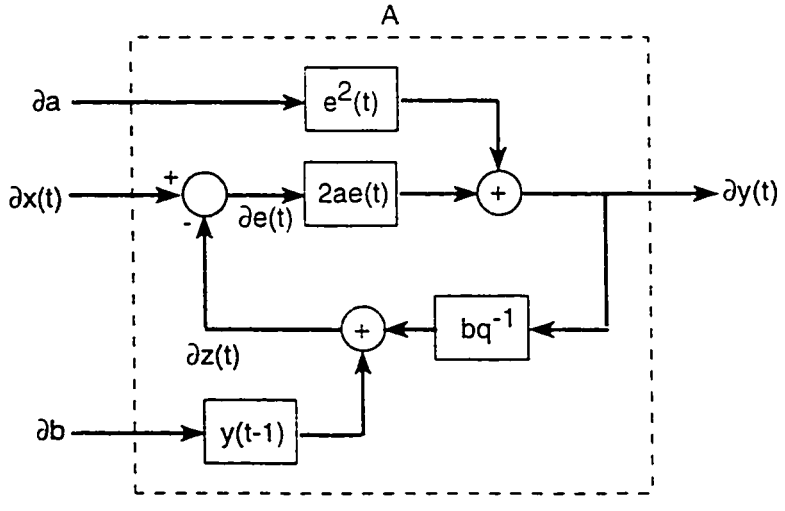


Figure 3.7: Linearized network for BPD computation

derivative of the output with respect to one of the inputs is given by the following equation.

$$\frac{\partial^A y}{\partial a} = \frac{\epsilon^2(t)}{1 + 2\epsilon(t)abq^{-1}} \tag{3.23}$$

The network whose BPD is to be computed can be linear or nonlinear, static or dynamic, ordered or unordered and it can be single or multiple input-output network.

If the network is ordered and static then the BPD reduces to the OPD. BPD is a convenient tool to compute gradients in the presence of complex interconnection of many blocks thus making it more attractive than DB or BPTT.

Due to the simplicity in calculation as compared to DB and BPTT, we will be using the BPD's to calculate the gradient for the weight update of the neural network.

Chapter 4

Observer Design

This chapter describes the standard observer design approaches for linear systems. Static observers and then variations of dynamic observer designs for dynamic systems are discussed. Kalman estimators and reduced order observers are also briefly reviewed.

The device which constructs an approximation of the state vector from the measured input and output is known as a state estimator or an observer. Observers play an important role in system control and diagnosis, as in reality, it is usually not possible to have an access to all the states of a system either due to the reason that the states are not accessible for direct measurement or that the number of measurable devices are limited.

In this chapter, we consider a linear system given by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{4.1}$$

where $x_k \in \Re^n$, $u_k \in \Re^r$ and $y_k \in \Re^m$.

Different types of state estimation techniques for such systems are now standardized in the text [50, 51]. One of these is the direct approach or static approach, in which the states are calculated from a sequence of inputs and outputs. The others are through the use of dynamic models (asymptotic or Luenberger observers). Reduced order observers are used if a part of the state vector is to be estimated. Kalman filtering theory can also be utilized to estimate the states of the system.

4.1 Static Observer

In the static observer design, the states of the system are directly calculated from the sequence of inputs and outputs. Considering the output equation of the given system of Eq 4.1, we have

$$\begin{aligned} y_{k-n+1} &= Cx_{k-n+1} \\ y_{k-n+2} &= Cx_{k-n+2} \\ &= CAx_{k-n+1} + CBu_{k-n+1} \end{aligned}$$

Continuing we get

$$y_k = CA^{n-1}x_{k-n+1} + CA^{n-2}Bu_{k-n+1} + \cdots + CBu_{k-1} \tag{4.2}$$

These equations can be concatenated together as follows

$$Y_n(k-n+1) = M_o x_{k-n+1} + \Omega U_{n-1}(k-n+1) \tag{4.3}$$

where $Y_n(k-n+1)$ is an output sequence of length n starting at the time instant $(k-n+1)$, as follows

$$Y_n(k-n+1) \doteq \begin{bmatrix} y_{k-n+1} \\ y_{k-n+2} \\ \vdots \\ y_k \end{bmatrix} \tag{4.4}$$

Further, $U_{n-1}(k-n+1)$ is an input sequence of length $n-1$ starting at the time instant $(k-n+1)$, i.e.

$$U_{n-1}(k-n+1) \doteq \begin{bmatrix} u_{k-n+1} \\ u_{k-n+2} \\ \vdots \\ u_k \end{bmatrix} \tag{4.5}$$

M_o is the observability matrix given by

$$M_o = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{4.6}$$

and Ω is given by

$$\Omega = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CB & 0 & \dots & 0 \\ \vdots & & & \\ CA^{n-2}B & CA^{n-3}B & \dots & CB \end{bmatrix} \tag{4.7}$$

The system is said to be observable, if the input sequence $U_{n-1}(k-n+1)$ and the output sequence $Y_n(k-n+1)$, for a finite k, are sufficient to determine uniquely the initial state x_{k-n+1} of the system. The initial state x_{k-n+1} can be obtained if and only if the observability matrix M_o of an nth order system is of full rank i.e. rank $M_o = n$.

Considering the system to be observable i.e. the observability matrix to be non-singular, the state x_{k-n+1} can be written as

$$\hat{x}_{k-n+1} = M_o^{-1} Y_n(k-n+1) - M_o^{-1} \Omega U_{n-1}(k-n+1)$$

Recursive use of Eq 4.1 will give the following equation

$$\hat{x}_k = A^{n-1} M_o^{-1} Y_n(k-n+1) + \Psi U_{n-1}(k-n+1) \tag{4.8}$$

where

$$\Psi = [\ A^{n-2}B \quad A^{n-3}B \quad \cdots \quad B\ ] - A^{n-1}M_o^{-1}\Omega \tag{4.9}$$

This equation shows that the state vector is a linear combination of the outputs $y_k, y_{k-1}, \ldots, y_{k-n+1}$ and of the inputs $u_{k-1}, u_{k-2}, \ldots, u_{k-n+1}$. The algorithm can be initiated after atmost n measurements of inputs and outputs. A static observer however is sensitive to the disturbances.

4.2 Dynamic Observers

One of the basic dynamic observers is the open loop observer. Consider the system given by the Eq 4.1 which is reproduced below.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned}$$

Let $\hat{x}$ be the estimates of the state vector x, the model of the observer is given by

$$\begin{aligned} \hat{x}_{k+1} &= A\hat{x}_k + Bu_k \\ \hat{y}_k &= C\hat{x}_k \end{aligned} \tag{4.10}$$

Same inputs are applied to the system and the model. If the initial conditions of the two are same then the output $\hat{x}_k$ will be identical to the actual state x_k. If

the initial states are different then $\hat{x}$ will converge to x only if the system of Eq 4.1 is asymptotically stable. However, any discrepancy between actual system matrices and the ones used in equation Eq 4.10 will yield unacceptable results. In reality such discrepancy cannot be avoided: an identified model is always different from the true system representation.

Closed loop observers are more common in use. These observers utilize both the inputs as well as the outputs of the system. Closed loop observers are more robust and are capable of handling modelling errors. Different methods for the design of observers are discussed below.

4.2.1 Method I

In this design the output of the model $\hat{y}_k$ is compared with the actual plant output y_k and the difference, termed as the (output) *error* is utilized to generate a correction term. The observer design is shown in Fig 4.1. Defining the error term e_k

$$\begin{aligned} e_k &= y_k - \hat{y}_k \\ &= C[x_k - \hat{x}_k] \end{aligned} \tag{4.11}$$

and the predicted value $\bar{x}_k$ as

$$\bar{x}_{k+1} = A\hat{x}_k + Bu_k \tag{4.12}$$

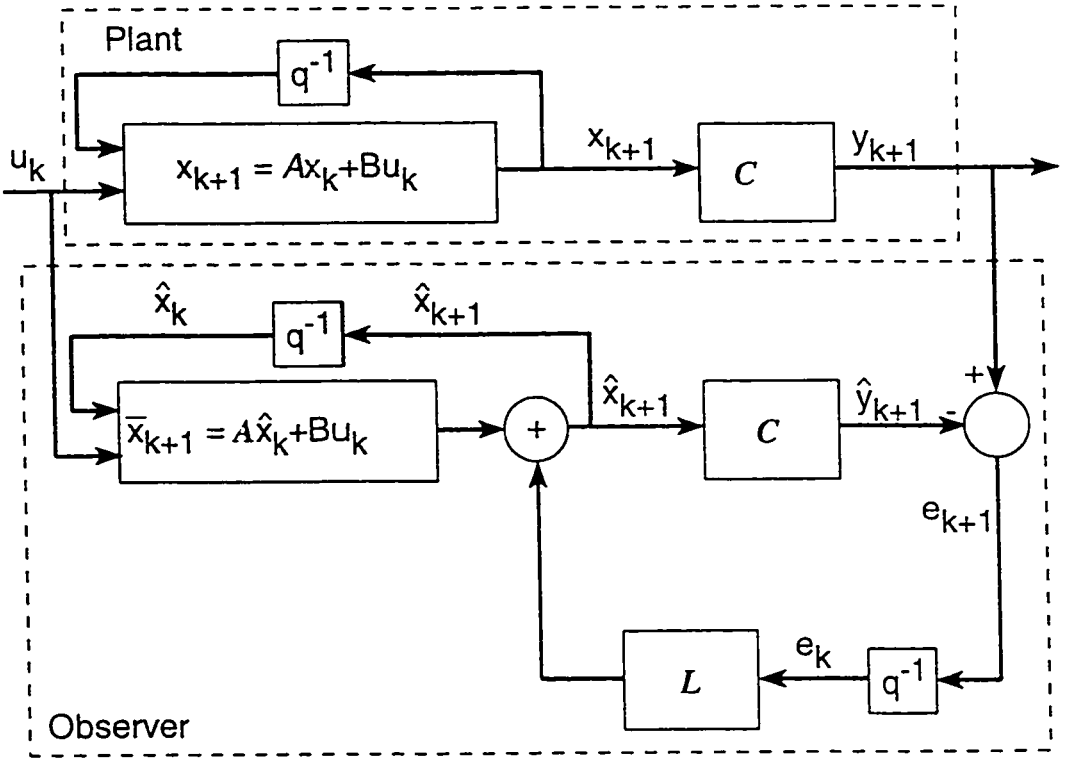


Figure 4.1: Linear observer method I

the observer equation can be written as

$$\hat{x}_{k+1} = \bar{x}_{k+1} + Le_k \tag{4.13}$$

Where L is the observer gain matrix. Define the reconstruction error $\tilde{x}$ as

$$\tilde{x} = x - \hat{x} \tag{4.14}$$

Subtracting Eq 4.13 from Eq 4.1, one gets

$$\begin{aligned} \tilde{x}_{k+1} &= A\tilde{x}_k - Le_k \\ &= (A - LC)\tilde{x}_k \end{aligned} \tag{4.15}$$

The gain matrix L is chosen such that the system is asymptotically stable and the reconstruction error converges to zero. This type of observer is often referred as *Prediction Observer*, as the observations at time $(k+1)$ are based on the output at time (k).

4.2.2 Method II

This is a variation of Method I. In Method I, the observer design has a delay as the new estimated state $\hat{x}_{k+1}$ depends only on the output values y_k. To overcome this delay, a modified observer design can be used as follows:

$$\begin{aligned} \hat{x}_{k+1/k+1} &= \hat{x}_{k+1/k} + Le_{k+1} \\ &= \hat{x}_{k+1/k} + L[y_{k+1} - C\hat{x}_{k+1/k}] \end{aligned} \tag{4.16}$$

where

$$\hat{x}_{k-1/k} = A\hat{x}_{k/k} + Bu_k \tag{4.17}$$

In the above $\hat{x}_{k+1/k}$ and $\hat{x}_{k+1/k+1}$ respectively represent the estimate of x_{k+1} based on the kth and $(k+1)$th output. Substituting Eq 4.17 and Eq 4.1 in Eq 4.16 gives

$$\begin{aligned} \hat{x}_{k+1/k+1} &= A\hat{x}_{k/k} + Bu_k + LCA[x_k - \hat{x}_{k/k}] \\ &= A\hat{x}_{k/k} + Bu_k + Le_{k+1} \end{aligned} \tag{4.18}$$

The reconstruction error is obtained by subtracting Eq 4.18 from Eq 4.1 as

$$\begin{aligned} \tilde{x}_{k+1/k+1} &= x_{k+1} - \hat{x}_{k+1/k+1} \\ &= (I - LC)A\tilde{x}_{k/k} \end{aligned} \tag{4.19}$$

In this scheme, L is chosen such that $(I - LC)$ have all eigen values inside the unit circle. This observer is referred as a *Current Observer*.

4.2.3 Method III

In this method a direct approach is taken to estimate the states of the dynamic system given by Eq 4.1. Based on the output equation of the system of Eq 4.1, define state z_k as an estimate of Tx_k given by the following equation

$$z_{k+1} = Fz_k + Gy_k + Hu_k \tag{4.20}$$

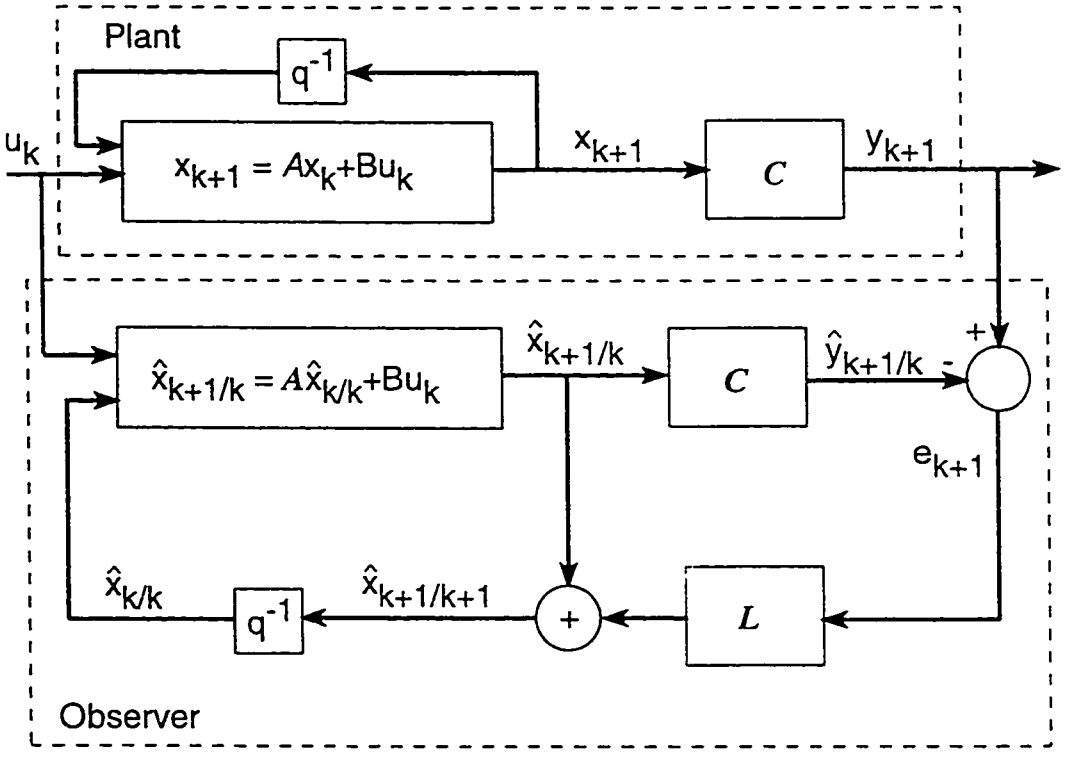


Figure 4.2: Linear observer method II

where $F \in \Re^{n\times n}$, $G \in \Re^{n\times m}$, $H \in \Re^{n\times r}$ and $T \in \Re^{n\times n}$ are all real constant matrices.

For any given x_0, z_0 and u_k, $z_k - Tx_k \to 0$ as $t \to \infty$, if and only if all the eigen values of F have magnitude less than 1 and the following equations are satisfied.

$$\begin{aligned} TA - FT &= GC \\ H &= TB \end{aligned} \tag{4.21}$$

Equation 4.20 gives an estimate of Tx_{k+1}. The estimate $\hat{x}_{k+1}$ can be written as

$$\begin{aligned} \hat{x}_{k+1} &= T^{-1}z_{k+1} \\ &= T^{-1}(Fz_k + Gy_k + Hu_k) \end{aligned} \tag{4.22}$$

To obtain the estimates $\hat{x}_{k+1}$, F is chosen such that all its eigen values have magnitude less than 1 and are distinct from that of A. A proper choice of G is then made such that the pair $\{F, G\}$ is controllable. Then Eq 4.21 is solved for T and the singularity of the matrix T so obtained, is checked. If T comes out to be singular then the process is repeated for a different value of F or G or both. Once a nonsingular T is obtained, H is obtained using the relation $H = TB$.

4.2.4 Reduced Order Observer

If in a given system, certain states are accurately measurable then these need not be estimated and only the remaining states have to be estimated. The resulting observer

is called the reduced order observer. However if the measurements are inaccurate or unreliable then a full state observer is preferable.

To design a reduced order observer we consider the n-dimensional system given by Eq 4.1 and assuming that output matrix C has a full row rank i.e. rank $C = m$. This implies that m states are available and $(n-m)$ states are to be estimated by the observer. Defining z_k as the estimate of Tx_k, where $T \in \Re^{(n-m)\times n}$ is a real constant matrix. Based on the direct approach of the full order observers (Method III), the design procedure for an $(n-m)$ dimensional observer is obtained as follows. Let

$$z_{k+1} = Fz_k + Gy_k + Hu_k \tag{4.23}$$

where $z \in \Re^{(n-m)}$, $F \in \Re^{(n-m)\times(n-m)}$, $G \in \Re^{(n-m)\times m}$ and $H \in \Re^{(n-m)\times r}$ are real constant matrices. F, G and T are designed in a similar way as described in the previous section. Once the matrix T is obtained then the matrix $P \in \Re^{n\times n}$ is given by

$$P = \begin{bmatrix} C \\ T \end{bmatrix} \tag{4.24}$$

The singularity of the matrix P is checked and the process is repeated with a different F or G or both until a nonsingular P is obtained. H is obtained by the relation $H = TB$.

The estimated states can be obtained by

$$\hat{x}_k = \begin{bmatrix} C \\ T \end{bmatrix}^{-1} \begin{bmatrix} y_k \\ z_k \end{bmatrix} \tag{4.25}$$

4.3 Kalman Estimator

Kalman estimator is an optimal mean square state estimation technique in the presence of noise. Consider the following plant equations

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + B_1 w_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{4.26}$$

where $x_k \in \Re^n$, $u_k \in \Re^r$ and $y_k \in \Re^m$. w_k and v_k are the plant and output noises respectively. Both w_k and v_k are assumed to be uncorrelated, zero mean white noise having Gaussian distributions with known covariances R_w and R_v respectively. Using $\tilde{x}$ from equation 4.14, we define the covariance matrix P_k as

$$P_k = E\{\tilde{x}_k . \tilde{x}_k^T\} \tag{4.27}$$

Kalman estimator minimizes the estimated covariance matrix P_k. The estimate $\hat{x}_k$ of x_k, the covariance matrix P_k and the Kalman gain K_k can be calculated by the following recursive equations

$$\hat{x}_k = \bar{x}_k + K_k[y_k - C\bar{x}_k]$$

$$\begin{aligned}
\bar{x}_{k+1} &= A\hat{x}_k + Bu_k \\
K_k &= \bar{P}_k C^T \left[C\bar{P}_k C^T + R_v\right]^{-1} \\
P_k &= \bar{P}_k - K_k C\bar{P}_k \\
&= (I - K_k C)\bar{P}_k \\
\bar{P}_{k+1} &= AP_k A^T + B_1 R_w B_1^T
\end{aligned} \tag{4.28}$$

where $\bar{P}_k$ is the covariance of the prediction errors:

$$\bar{P}_k = E\{[x_k - \bar{x}_k]\,[x_k - \bar{x}_k]^T\} \tag{4.29}$$

Application of Kalman filters as state estimators in state feed-back control systems may result in systems that are nonrobust [52]. The covariances of the noise must be known and in case of erroneous models the performance of the Kalman filter may deteriorate.

Chapter 5

Neural Observers

In this chapter, the techniques of observer design for linear systems described in the previous chapter are extended to the design of neural observers for nonlinear systems. First a static observer scheme as developed by Narendra [42] is presented. Then design of dynamic neural observer is considered. Six different schemes are developed in this category and each scheme is discussed in detail.

The linear observer design techniques are now matured and they appear in the standard texts. However most of the practical systems are nonlinear and therefore nonlinear observer design is one of the most important area in systems and control theory. Research work in this field is only in the developing stage. Most of the nonlinear designs are based on linearization or quasi-linearization of the system and application of the techniques of linear systems to the linearized models. Linearization

of a system produces an approximation of the plant around an operating point. This approximation, also called the locally linearized model, works satisfactorily in the neighborhood of the operating point. But with a significant change in the operating point, the parameters of the linearized model also change. Two different approaches are in practice to rectify this problem. In the first approach a family of fixed number of linearized models around different operating points are considered and a neighborhood for each operating point is specified where the system performs satisfactorily. The second approach considers a continuous change of locally linearized model with the operating point is considered [53]. Our developments are based on the second approach.

Through out this chapter a nonlinear system described by the following equations is considered:

$$\begin{aligned} x_{k+1} &= f(x_k, u_k) \\ y_k &= h(x_k) \end{aligned} \tag{5.1}$$

where $x \in \Re^n$, $u \in \Re^r$ and $y \in \Re^m$, $f : \Re^{n+r} \rightarrow \Re^n$ and $h : \Re^n \rightarrow \Re^m$. The locally linearized model of the system of Eq 5.1 around an operating point can be written as

$$\begin{aligned} \delta x_{k+1} &= A_k \delta x_k + B_k \delta u_k \\ \delta y_k &= C_k \delta x_k \end{aligned} \tag{5.2}$$

where A_k, B_k and C_k are the Jacobians of the linearized system, given by

$$\begin{aligned} A_k &= \frac{\partial f(x_k, u_k)}{\partial x_k} \\ B_k &= \frac{\partial f(x_k, u_k)}{\partial u_k} \\ C_k &= \frac{\partial h(x_k)}{\partial y_k} \end{aligned} \tag{5.3}$$

However, the above locally linearized model is only valid for small signal change. A generalization of the locally linearized model (Eq 5.2) for the system of Eq 5.1 is given by Ahmed [54] as

$$\begin{aligned} \tilde{x}_{k+1} &= A(I_k)\tilde{x}_k + B(I_k)\tilde{u}_k \\ \tilde{y}_k &= C(I_k)\tilde{x}_k \end{aligned} \tag{5.4}$$

$$\text{where} \qquad \begin{aligned} I_k &= [y_k, x_{k+1}, O_k^T, \tilde{O}_k^T]^T \\ O_k &= [x_k^T, u_k^T]^T \end{aligned}$$

and the tilde ($\sim$) indicates the incremental values. The elements of A, B and C are defined as

$$\begin{aligned} a_{ij}(I_k) &= \left(\frac{\partial f_i(O_k)}{\partial x_{j(k)}}\tilde{x}_{i(k+1)}\right) \Big/ M_{i(k)} \\ b_{ij}(I_k) &= \left(\frac{\partial f_i(O_k)}{\partial u_{j(k)}}\tilde{x}_{i(k+1)}\right) \Big/ M_{i(k)} \end{aligned}$$

$$\text{and} \qquad c_{ij}(I_k) = \left(\frac{\partial h_i(O_k)}{\partial x_{j(k)}}\tilde{y}_{i(k)}\right) \Big/ N_{i(k)} \tag{5.5}$$

where $M_{i(k)}$ and $N_{i(k)}$ are given as

$$M_{i(k)} = \sum_{j=1}^{n} \frac{\partial f_i(O_k)}{\partial x_{j(k)}}\tilde{x}_{j(k)} + \sum_{j=1}^{r} \frac{\partial f_i(O_k)}{\partial u_{j(k)}}\tilde{u}_{j(k)}$$

$$N_{i(k)} = \sum_{j=1}^{n} \frac{\partial h_i(O_k)}{\partial x_{j(k)}} \tilde{x}_{j(k)}$$

This generalization (Eq 5.4) collapses to locally linearized model (Eq 5.2) for small signal variations: as $\tilde{x}_{i(k)}$ and $\tilde{u}_{i(k)}$ respectively approach $dx_{i(k)}$ and $du_{i(k)}$ resulting in $M_{i(k)}$ and $N_{i(k)}$ respectively to approach $dx_{i(k+1)}$ and $dy_{i(k)}$. By direct substitution it can be verified that the above generalization is valid for arbitrary change in state, input and output variables.

It can be observed that the coefficients of the generalized linear representation (Eq 5.4) of the nonlinear system are not only dependent on the operating point of the system but also on the signal increments at each time step. Therefore a family of linear models may exist for each operating point of the system and an observer based on all these linear models at each operating point will perform better than the one based on locally linearized models.

The tremendous capability of neural networks for function approximations can be utilized to design such observers for nonlinear systems. A neural network based observer can be directly trained to perform satisfactorily for a family of all possible operating points of the system. Utilizing this concept, we are proposing dynamic neural observers. These schemes utilize the generalized linearization concept and are trivial extensions to the approaches of linear dynamic observers, given in the previous chapter. Neural networks are used to approximate the nonlinear functions and/or observer gains. The trained neural networks of the proposed schemes may be

considered as a concatenation of all possible linear observers at all operating points.

Although Narendra [42] also proposed a scheme for the design of a neural observer, but his scheme is based on static observer design theory utilizing the input and output sequences of the system. Such an observer does not exploit the error dynamics making the observer nonrobust i.e. in case of erroneous models the performance of the observer deteriorates. Where as the proposed schemes are relatively robust having better performance in the presence of disturbances and model variations.

The amount of computation involved in the design of dynamic neural observers is mainly during the training of the observer. Once the observer is trained the computations involved are trivial. This gives the neural observer an edge over other techniques like extended Kalman filter which requires considerable computations at the implementation step.

Static observer and a variation of different schemes of dynamic neural observers are given in the following sections.

5.1 Static Neural Observer

Based on the static observer design for linear systems, Levin and Narendra [42] trained a neural network to observe the nonlinear system states. Consider a nonlinear system described by Eq 5.1. It is assumed that the functions f and h are known.

Based on the properties of f and h, the conditions for local observability of the

system is derived. Given an equilibrium state, it is proved in [41, 42] that there exists a region around that equilibrium state, any state belonging to it can be uniquely determined by probing the system with an input sequence of sufficient length. It is shown that if the linearized system given by Eq 5.2 around the equilibrium is observable then the given nonlinear system is locally strongly observable.

Defining $Y_i(k)$ and $U_i(k)$ as sequences of outputs and inputs as given by Eq 4.4 and Eq 4.5 respectively, it may be observed that the states and outputs of Eq 5.1 at the ith time instant can be computed from the knowledge of the initial states and the input sequence by iteratively using Eq 5.1. Therefore one may express x_{k+i} in terms of x_k and $U_i(k)$ as

$$x_{k+i} \equiv F_i(x_k, U_i(k)) \tag{5.6}$$

Similarly the output y_{k+i} can be written as

$$\begin{aligned} y_{k+i} &\equiv h(x_{k+i}) \\ &\equiv \tilde{h}(x_k, U_i(k)) \end{aligned} \tag{5.7}$$

where $F_i : \Re^{n+i} \rightarrow \Re^n$ and $\tilde{h} : \Re^{n+i} \rightarrow \Re^m$. The concatenation of outputs of Eq 5.1 as a function of initial states and input after rearranging the indices is given by

$$Y_i(k) \equiv H_i(x_k, U_{i-1}(k)) \tag{5.8}$$

The Jacobian of $Y_i(k)$ with respect to x_k, evaluated at the operating point, is the observability matrix M_o (see Eq 4.6) of the linearized model given by Eq 5.2. If M_o

is of full rank i.e. the linearized model given by Eq 5.2 is observable, there exists a neighborhood of the operating point on which H_i is invertable. Let $\tilde{\Phi}$ be the inverse of H_i, then we have

$$x_k = \tilde{\Phi}\left(Y_i(k), U_{i-1}(k)\right) \tag{5.9}$$

As mentioned in the linear case, any sequence of length n, n being the order of the system, will be sufficient to determine the states uniquely. Since we are interested in the state of the system at time instant k after the probing input is applied, we rearrange the indices to get

$$x_{k+n-1} = F_{n-1}\left(x_k, U_{n-1}(k)\right) \tag{5.10}$$

If the linearized model is observable, there exist a function Φ such that

$$x_{k+n-1} = \Phi\left(Y_n(k), U_{n-1}(k)\right) \tag{5.11}$$

after rearranging indices we get the observer equation as

$$\hat{x}_k = \Phi\left(Y_n(k-n+1), U_{n-1}(k-n+1)\right) \tag{5.12}$$

A feed-forward network NN_Φ is trained to emulate the nonlinear function Φ. Since f and h are assumed to be known, the training of the network can be done offline. At each instant of time the inputs to the neural network are a sequence of the past $n-1$ system inputs and the n system outputs. The output of the network are the estimated states $\hat{x}_k$. This is compared with the state of the simulated system and the

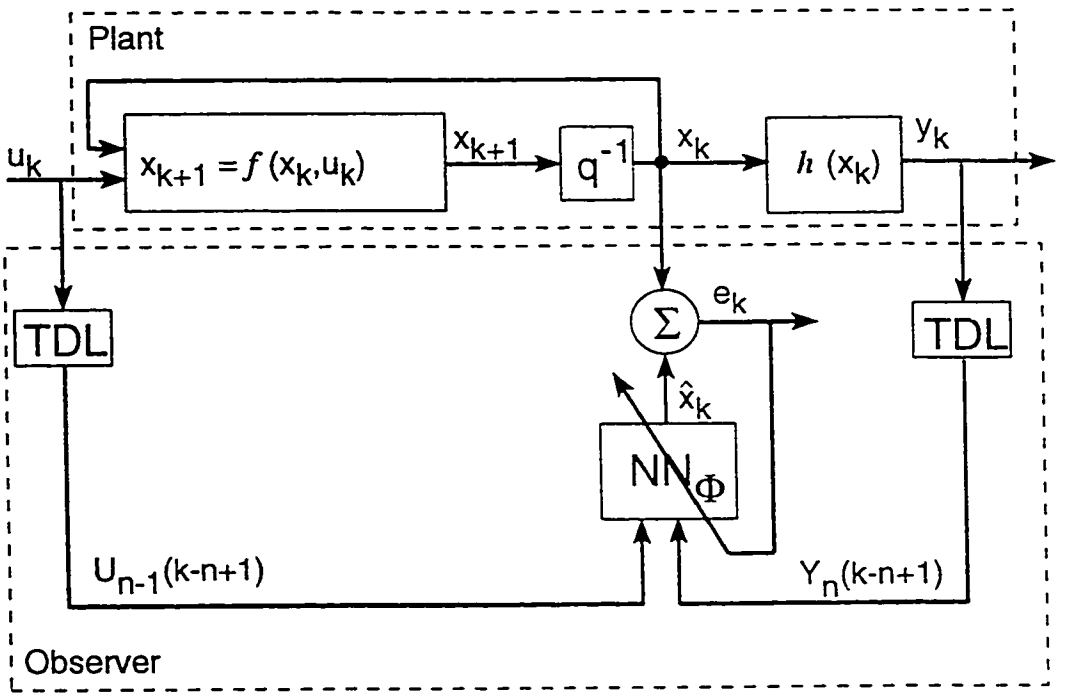


Figure 5.1: Static state estimation scheme

error is given by

$$e_k = x_k - NN_\Phi(Y_n(k-n+1), U_{n-1}(k-n+1)) \tag{5.13}$$

The network is trained through backpropagation algorithm by minimizing the error norm as described in the previous chapter . The block diagram in Fig 5.1 shows the setup for state estimation of the system by static neural observer. TDL stands for tapped time delay which gives the delayed inputs and outputs.

5.2 Dynamic Neural Observers

In this section the dynamic observer theory for linear systems is extended to the nonlinear systems. Dynamic observers have many advantages over static observers as discussed in the previous chapter. The first four schemes are developed from Method I and II described in sections 4.2.1 and 4.2.2. These schemes are based on a predictor-corrector format. First a predictive value of the state is obtained from the model of the plant and then a correction term based on the recent measurements is introduced to it. The approach resembles the Kalman estimator but the correction term required is obtained by the neural network as will be explained later for each scheme.

A more direct approach, scheme 5 is an extension of Method III (section 4.2.3) of the previous chapter. The input and the output of the plant along with the observed states at (k)th instant are fed to the neural network which in turn gives the observed state for the $(k+1)$th instant. A reduced order nonlinear observer is also presented that imitates the reduced order linear observer described in section 4.2.4.

In the subsequent developments, a nonlinear system described by Eq 5.1 is considered. It is assumed that the system is locally observable, and that f and h are known. In all the proposed schemes, multi-layer feed-forward neural networks (MFNN) are used for plant nonlinearity and/or observer gain approximations. The observers are trained offline in a close loop configuration that incorporates the model of the system.

5.2.1 Scheme 1

This scheme is an extension of the *Prediction* observer, described in section 4.2.1 where the gain matrix is approximated using the neural networks. The block diagram for scheme 1 of the neural observer is shown in Fig 5.2. Following Method I of section 4.2.1, in this scheme a predicted value of the state $\bar{x}_k$ is obtained using the model of the system. A corrected state estimate is then obtained from the predicted estimate and the output error. The equations for the predicted state $\bar{x}_k$ and the estimated output $\hat{y}_k$ are given by the following equations.

$$\bar{x}_{k-1} = f(\hat{x}_k, u_k) \tag{5.14}$$

$$\hat{y}_k = h(\hat{x}_k) \tag{5.15}$$

The output of the plant and the output of the model are compared to get the output error e_k, given by the following relation

$$e_k = y_k - \hat{y}_k \tag{5.16}$$

The error term, the predicted state and the previous estimates are used to get the corrected state estimate as given by the following relation

$$\hat{x}_{k+1} = \bar{x}_{k+1} + g(\hat{x}_k, e_k) \tag{5.17}$$

Where the nonlinear function $g(.)$ is to be determined based on optimizing a performance index. The function $g(.)$ although unknown at this stage, must conform to the

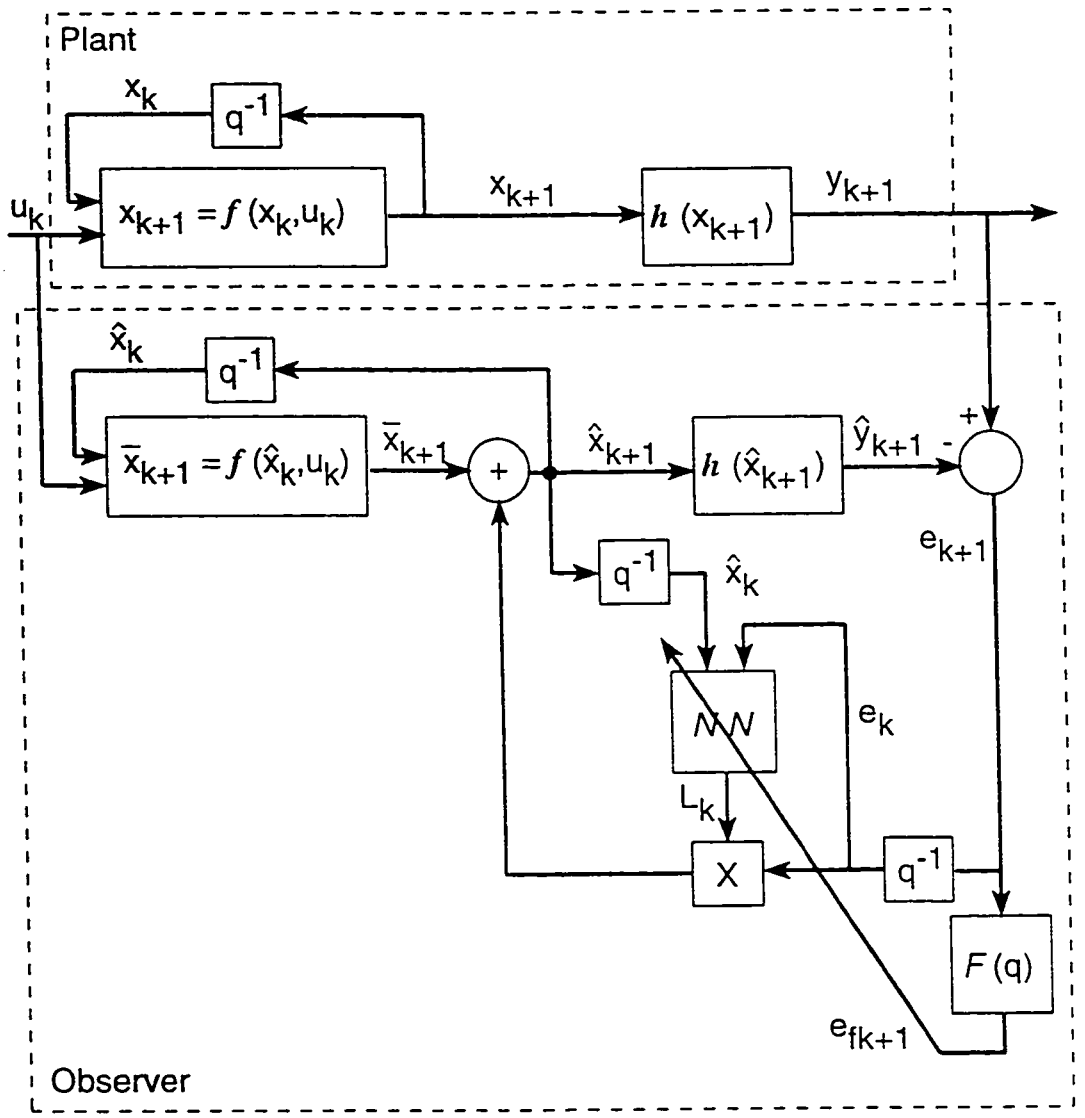


Figure 5.2: Dynamic state estimation scheme 1

following requirement: When the output error $\epsilon_k = 0$, the corrected estimate $\hat{x}_{k+1}$ should be equal to the predicted states $\bar{x}_{k+1}$, i.e.

$$\hat{x}_{k+1} = \bar{x}_{k+1} + g(\hat{x}_k, 0) = \bar{x}_{k+1} \tag{5.18}$$

$$\text{or} \qquad g(\hat{x}_k, 0) = 0 \tag{5.19}$$

Expanding $g(\hat{x}_k, \epsilon_k)$ about $g(\hat{x}_k, 0)$ by a Taylor series, one gets

$$\begin{aligned} g(\hat{x}_k, \epsilon_k) &= g(\hat{x}_k, 0) + \left(\frac{\partial g}{\partial e_k}|_{e_k=0}\right)\epsilon_k + \left(\frac{1\partial^2 g}{2!\partial e_k^2}|_{e_k=0}\right)e_k^2 + \ldots \\ g(\hat{x}_k, \epsilon_k) &= \left[\left(\frac{\partial g}{\partial e_k}|_{e_k=0}\right) + \left(\frac{1\partial^2 g}{2!\partial e_k^2}|_{e_k=0}\right)e_k + \ldots\right]e_k \end{aligned} \tag{5.20}$$

Where the last identity follows from Eq 5.18. Equation 5.20 can be written as

$$g(\hat{x}_k, e_k) = L(\hat{x}_k, e_k)e_k \tag{5.21}$$

where $L(\hat{x}_k, e_k)$: $\Re^{m+n} \rightarrow \Re^n$ while $e \in \Re^m$ and $x \in \Re^n$. Thus we get the final correction equation as

$$\hat{x}_{k+1} = \bar{x}_{k+1} + L(\hat{x}_k, \epsilon_k)e_k \tag{5.22}$$

A multi-layer feed-forward neural network (MFNN) will be used to represent the function $L(\hat{x}_k, e_k)$. As shown in the block diagram Fig 5.2, the input to the neural network is $\hat{x}_k$ and ϵ_k and the output of the neural network is $L(\hat{x}_k, e_k)$, which pre-multiplies the error ϵ_k to produce the final correction term. For the training of the neural network, the criterion function can be chosen as

$$J = \frac{1}{2}\sum_{k=1}^{n} \|e_{fk-1}\|^2 \tag{5.23}$$

where e_{fk+1} is the filtered error obtained by passing the error e_{k+1} through a moving average filter having all roots within the unit circle. The training scheme can be summarized as follows:

Network Training: Initially the neural network weights are chosen as random. The plant and the model are initialized with a set of randomly chosen nonidentical initial conditions and the output of the plant and the model are calculated. At each time instant the output of the model is compared with the plant output to get the output error e_{k+1} and the error so obtained is filtered to get e_{fk+1}. In order to calculate the gradients the BPD technique is employed. The linearized block diagram is shown in Fig 5.3. In this figure the derivatives with superscript "+" indicates the ordered partial derivatives of the neural network [37]. After the update of weights, the process is repeated until a prescribed error level is achieved. The weight update equation is given as

$$W_{k+1} = W_k + \eta \sum_{k=1}^{n} \Gamma_{k+1} e_{fk+1} \tag{5.24}$$

$$\text{where} \qquad \Gamma_{k+1} = \left(\frac{\partial \hat{y}_{fk+1}}{\partial W} \right)^T \tag{5.25}$$

η is the learning rate. The choice of the learning rate will be defined later. After simplification of the linearized network, by employing the block partial derivatives and Mason's rule, the following equations are obtained for the gradient

$$\frac{\partial \hat{y}_{fk+1}}{\partial W} = F C_k S_{k+1} \tag{5.26}$$

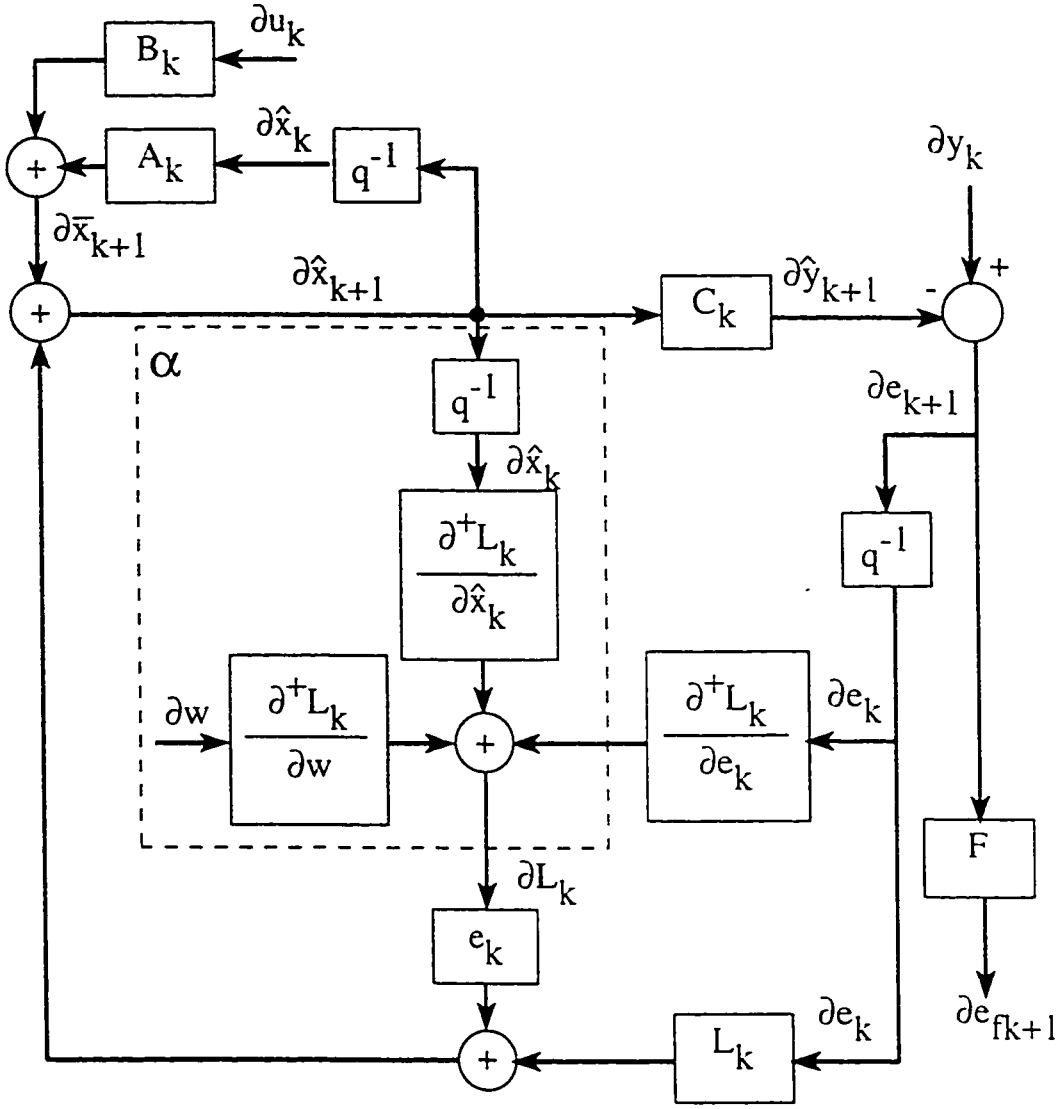


Figure 5.3: Linearized diagram for scheme 1

where F is a moving average filter as shown in Fig 5.1. This filter is used to obtain the filtered error as follows:

$$\begin{aligned} e_{fk} &= Fe_k \\ F &\equiv F(q^{-1}) = 1 + \beta_1 q^{-1} + \beta_2 q^{-2} \end{aligned} \tag{5.27}$$

with all the roots of $1 + \beta_1 q^{-1} + \beta_2 q^{-2}$ within the unit circle. S_{k+1} is defined as

$$\begin{aligned} S_{k+1} &= \frac{\partial^\alpha \hat{x}_{k+1}}{\partial W} \\ &= HS_k + e_k \frac{\partial^+ L_k}{\partial W} \end{aligned} \tag{5.28}$$

H is given by the following equation

$$H = e_k \frac{\partial^+ L_k}{\partial \hat{x}_k} - \left[L_k + e_k \frac{\partial^+ L_k}{\partial e_k} \right] C_k + A_k \tag{5.29}$$

A_k and C_k represent the locally linearized plant model at the operating point and are given as

$$\begin{aligned} A_k &= \frac{\partial \bar{x}_{k+1}}{\partial \hat{x}_k} \\ C_k &= \frac{\partial \hat{y}_k}{\partial \hat{x}_k} \end{aligned} \tag{5.30}$$

5.2.2 Scheme 2

This scheme depicted in Fig 5.4 is an alternate form of scheme 1. This scheme directly minimizes the state estimation error. The criterion function in this scheme is the sum

squares state error as against the filtered output error of scheme 1. The equations that describe the state error and the criterion function are given as follows

$$\tilde{x}_k = x_k - \hat{x}_k \tag{5.31}$$

$$J = \frac{1}{2}\sum_{k=1}^{n} \|\tilde{x}_{fk+1}\|^2 \tag{5.32}$$

where

$$\tilde{x}_{fk+1} = (I - A_0 q^{-1})\tilde{x}_{k+1} \tag{5.33}$$

While A_0 is defined as the observer matrix of the linearized model with the specified observer eigen values. The matrix can also be taken as diagonal. The weight update equation for this scheme is obtained as

$$W_{k+1} = W_k + \eta \sum_{k=1}^{n} \Gamma_{k+1}\tilde{x}_{fk+1} \tag{5.34}$$

$$\text{where} \qquad \Gamma_{k-1} = \left(\frac{\partial \hat{x}_{fk+1}}{\partial W}\right)^T \tag{5.35}$$

Using the BPD technique an expression for this sensitivity derivative can be obtained as

$$\frac{\partial \hat{x}_{fk+1}}{\partial W} = S_{k-1} - A_0 S_k \tag{5.36}$$

where S_k is given in Eq 5.28.

5.2.3 Scheme 3

This scheme is a direct extension of method II of linear dynamic observer discussed in section 4.2.2. The block diagram for this scheme is given in the Fig 5.5. In this

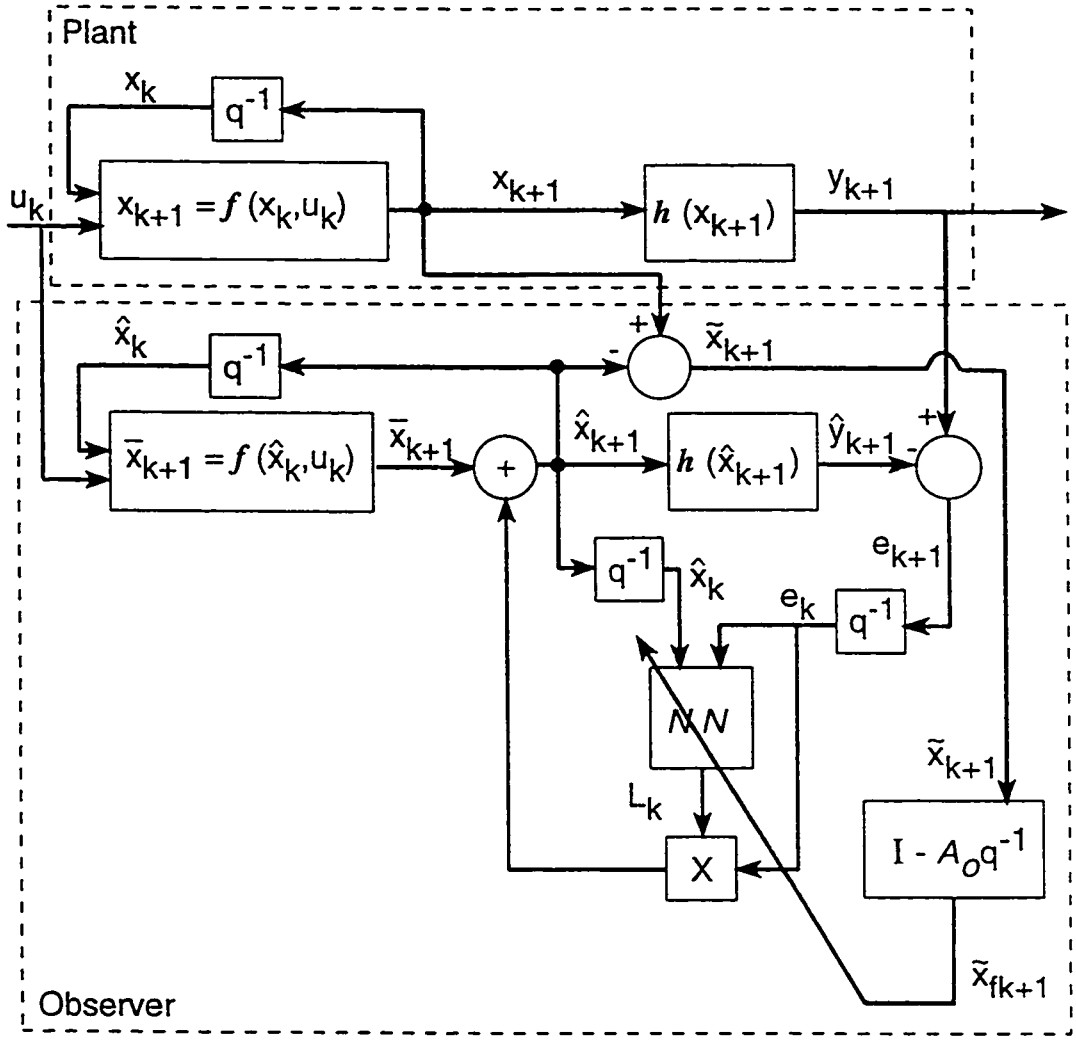


Figure 5.4: Dynamic state estimation scheme 2

scheme, with a given state $\hat{x}_{k/k}$, the predicted state $\hat{x}_{k+1/k}$ and the predicted output $\hat{y}_{k+1/k}$ are first obtained by using the following equations

$$\begin{aligned} \hat{x}_{k+1/k} &= f(\hat{x}_{k/k}, u_k) \\ \hat{y}_{k+1/k} &= h(\hat{x}_{k+1/k}) \end{aligned} \tag{5.37}$$

In the correction step, the output of the plant and the output of the model are compared and the error e_k, is generated as follows

$$e_{k+1} = y_{k+1} - \hat{y}_{k+1/k} \tag{5.38}$$

The error term and the predicted states are used to get a corrected state estimate by using the following general form

$$\hat{x}_{k+1/k+1} = g(\hat{x}_{k+1/k}, e_{k+1}) \tag{5.39}$$

After conforming to the same conditions as given in scheme 1 that is if $e_{k+1} = 0$, then the estimated states $\hat{x}_{k+1/k+1}$ should be equal to the predicted states $\hat{x}_{k+1/k}$, the equation for $g(\hat{x}_{k+1/k}, e_{k+1})$ can be written as

$$g(\hat{x}_{k+1/k}, e_{k+1}) = \hat{x}_{k+1/k} + L(\hat{x}_{k+1/k}, e_{k+1})e_{k+1} \tag{5.40}$$

where $L(\hat{x}_{k+1/k}, e_{k+1})$ is defined as a nonlinear function: $\Re^{n+m} \rightarrow \Re^n$. The above is a direct extension of observer equation for linear system given in Eq 4.18 and the final correction equation is given by

$$\hat{x}_{k+1/k+1} = \hat{x}_{k+1/k} + L(\hat{x}_{k+1/k}, e_{k+1})e_{k+1} \tag{5.41}$$

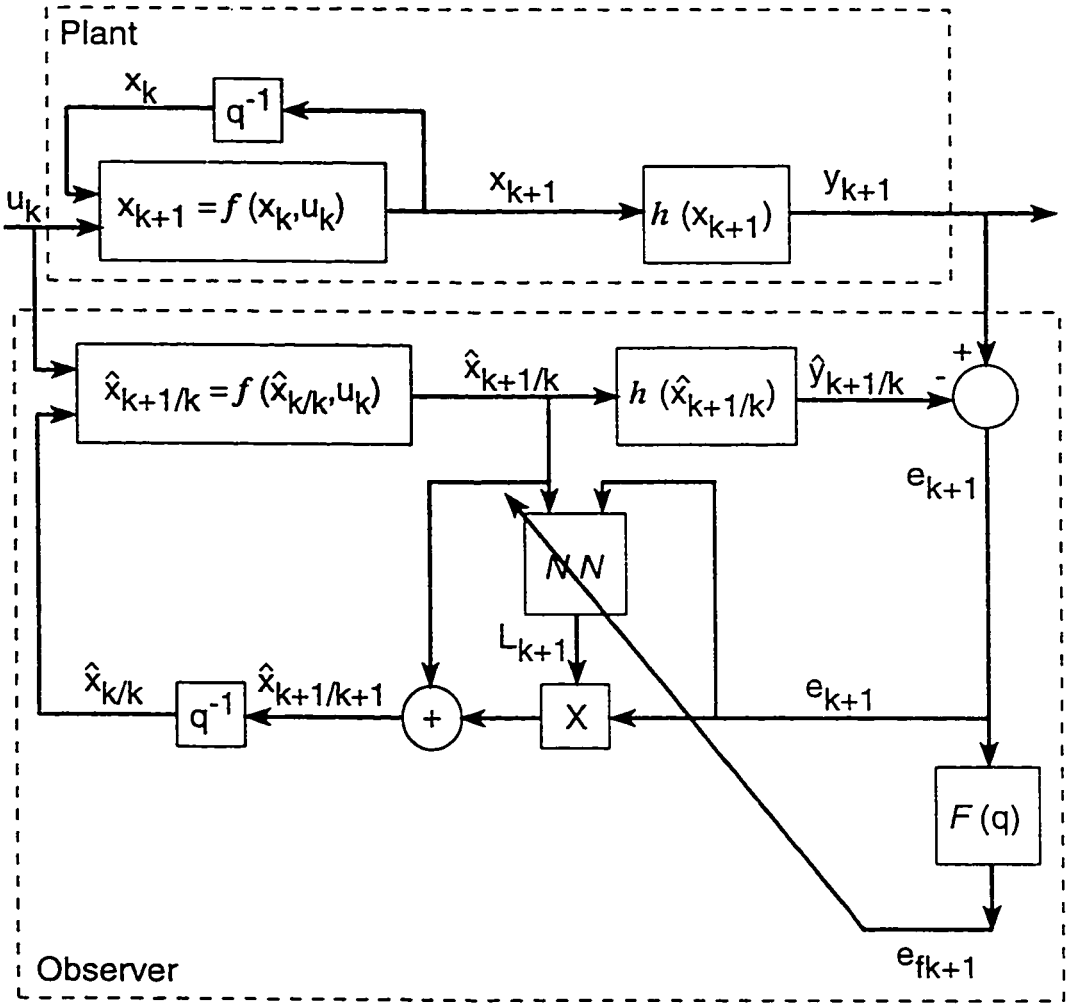


Figure 5.5: Dynamic state estimation scheme 3

Like scheme 1, a multi-layered feed-forward neural network (MFNN) is used to represent the function $L(\hat{x}_{k+1/k}, e_{k+1})$ as shown in the Fig 5.5. The predicted states $\hat{x}_{k+1/k}$ and the error e_{k+1} are the input to the neural network and the output of the neural network, $L(\hat{x}_{k+1/k}, e_{k+1})$, is passed through a special output block which guarantees that if the error is zero then no update to the predicted states occur. The criterion function for the training of the neural network is given as

$$J = \frac{1}{2}\sum_{k=1}^{n} \|e_{fk+1}\|^2 \tag{5.42}$$

The model is initialized in the same approach as described in scheme 1. The linearized model of the above configuration is shown in the Fig 5.6. This linearization is used to calculate the sensitivity derivatives which are required for the weight update of the neural network. BPD technique is used to calculate the sensitivity derivatives as shown in the linearized diagram. The ordered partial derivatives shown in the diagram are obtained by back-propagating ones in the neural network and the weight update equation can be written as

$$\begin{aligned} W_{k+1} &= W_k + \eta \sum_{k=1}^{n} \Gamma_{k+1} e_{fk+1} \\ \text{where} \qquad \Gamma_{k+1} &= \left(\frac{\partial \hat{y}_{fk+1}}{\partial W}\right)^T \end{aligned} \tag{5.43}$$

Γ_{k+1} is obtained by solving the following equations which are obtained by linearizing the network shown and applying the same procedure as described in scheme 1.

$$\frac{\partial \hat{y}_{fk+1}}{\partial W} = FC_k A_k \frac{\partial \hat{x}_{k/k}}{\partial W}$$

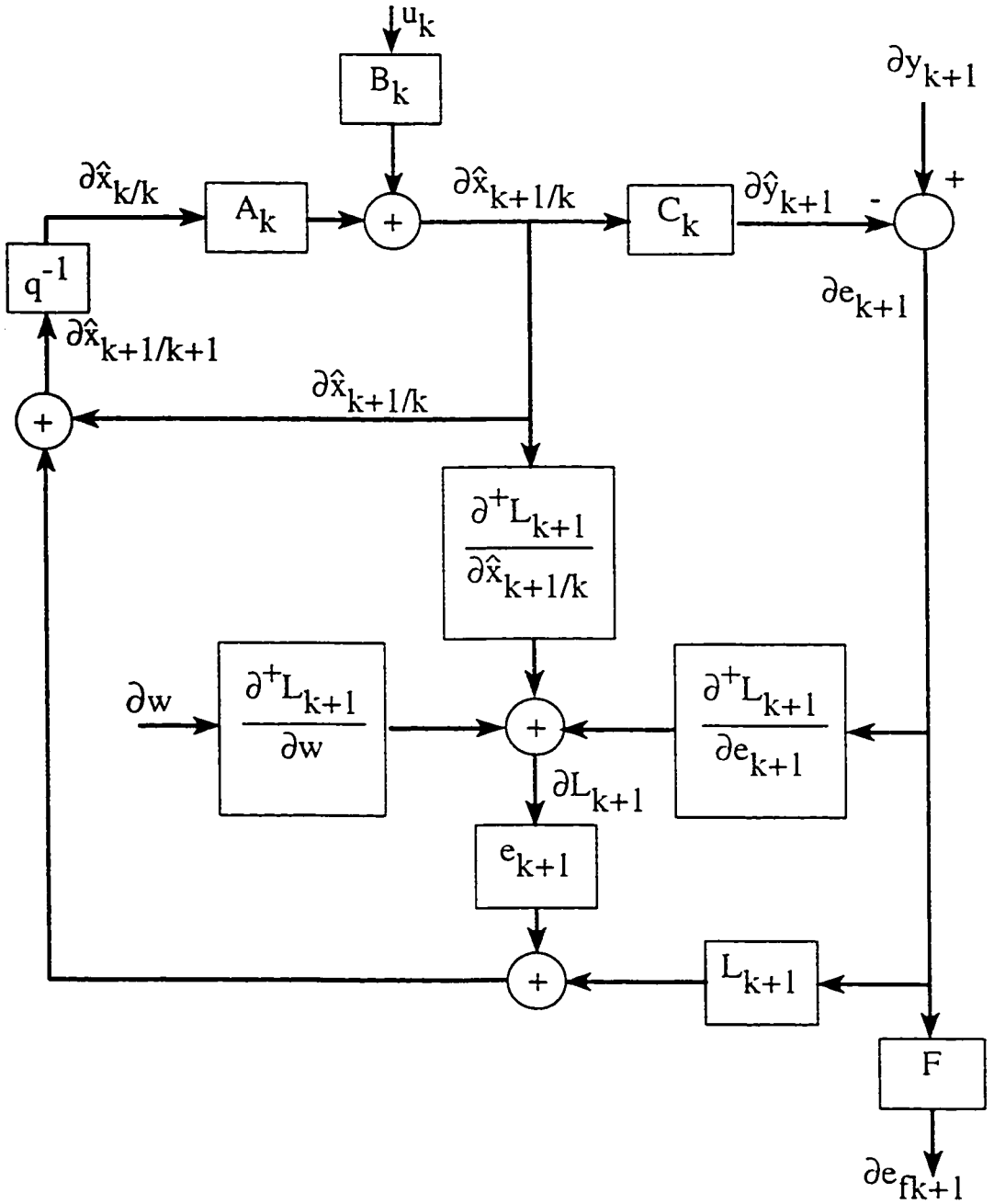


Figure 5.6: Linearized diagram for scheme 3

$$\frac{\partial \hat{x}_{k+1/k+1}}{\partial W} = H A_k \frac{\partial \hat{x}_{k/k}}{\partial W} + e_{k+1} \frac{\partial^+ L_{k+1}}{\partial W} \tag{5.44}$$

where H is given by

$$H = I + e_{k+1} \frac{\partial^+ L_{k+1}}{\partial x_{k+1/k}} - \left[L_{k+1} + e_{k+1} \frac{\partial^+ L_{k+1}}{\partial e_{k+1}} \right] C_k \tag{5.45}$$

A_k and C_k are the same as described in the Eq 5.30. An adaptive learning rate is taken for the training of the neural network as will be discussed in the next chapter.

5.2.4 Scheme 4

This scheme is a modified version of scheme 3, developed in the same spirit as scheme 2 was developed from scheme 1. The modification is that in place of output error e_k, state error $\tilde{x}_k$ is calculated and the criterion function is chosen to minimize the state error. The scheme is given in Fig 5.7 and the equations for state error and the criterion function can be written as

$$\tilde{x}_k = x_k - \hat{x}_{k/k} \tag{5.46}$$

$$J = \frac{1}{2} \sum_{k=1}^{n} \| \tilde{x}_{fk+1} \|^2 \tag{5.47}$$

where $\tilde{x}_{fk}$ is given by

$$\tilde{x}_{fk+1} = (I - A_0 q^{-1}) \tilde{x}_{k+1} \tag{5.48}$$

The algorithm to calculate the gradient for weight update is given as

$$W_{k+1} = W_k + \eta \sum_{k=1}^{n} \Gamma_{k+1} \tilde{x}_{fk+1}$$

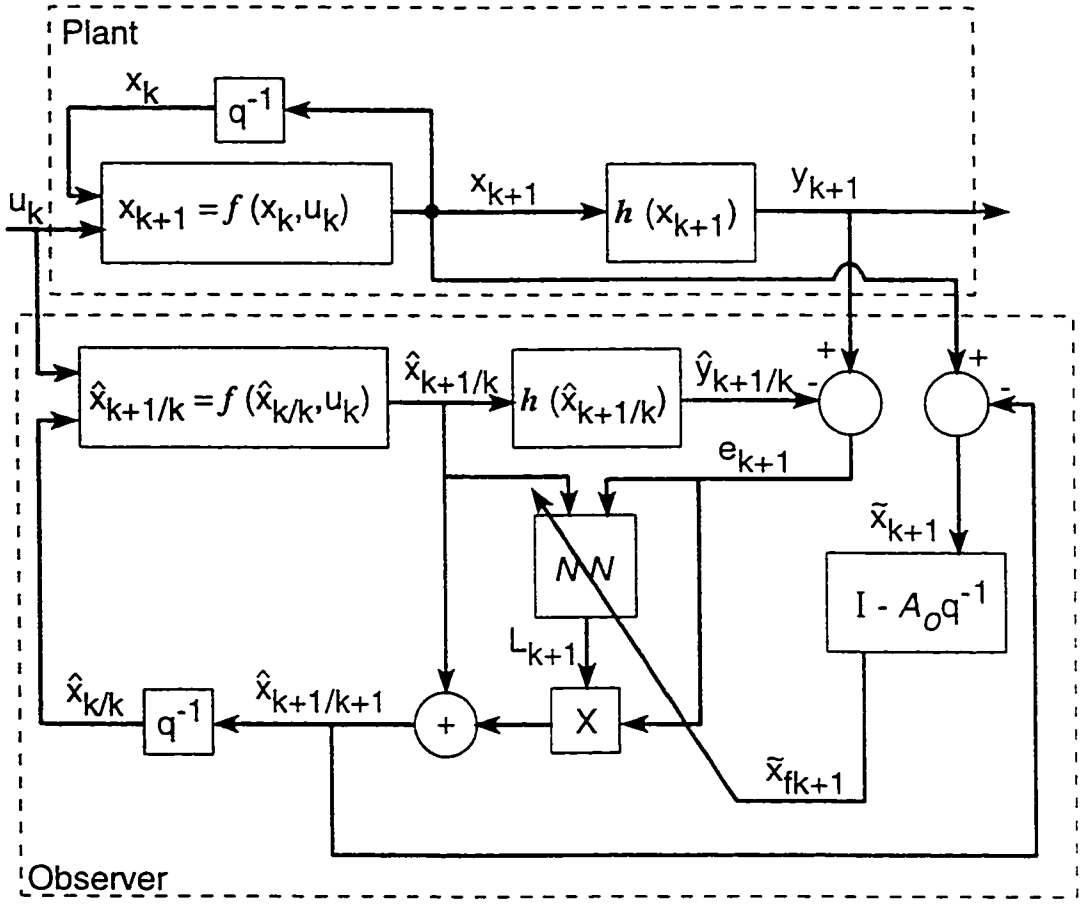


Figure 5.7: Dynamic state estimation scheme 4

$$\text{where} \qquad \Gamma_{k+1} = \left(\frac{\partial \hat{x}_{fk+1}}{\partial W}\right)^T \tag{5.49}$$

The value for Γ_{k+1} is calculated by the following equations

$$\frac{\partial \hat{x}_{fk+1}}{\partial W} = \frac{\partial \hat{x}_{k-1/k+1}}{\partial W} - A_0 \frac{\partial \hat{x}_{k/k}}{\partial W} \tag{5.50}$$

$$\frac{\partial \hat{x}_{k+1/k+1}}{\partial W} = H A_k \frac{\partial \hat{x}_{k/k}}{\partial W} + e_{k+1} \frac{\partial^+ L_{k+1}}{\partial W} \tag{5.51}$$

where H is the same as developed in scheme 3 and is given by equation Eq 5.45.

5.2.5 Scheme 5

This scheme is developed on the basis of a more a direct approach (Method III section 4.2.3). The block diagram of this scheme is shown in the Fig 5.8. The state observer equations for a linear system as given by Eq 4.22 is extended to a nonlinear system and is given by

$$\hat{x}_{k+1} = L(\hat{x}_k, y_{k+1}, u_k) \tag{5.52}$$

The nonlinear function L will be approximated by a neural network. The input and the output of the plant along with the estimated states at the previous instant are taken as the input to the neural network. The output of the neural network are the estimated states of the system. These estimated states are compared with the actual states of the system and the error generated is filtered and used to train the neural network. The criterion function for this scheme is given by Eq 5.53 and the weight

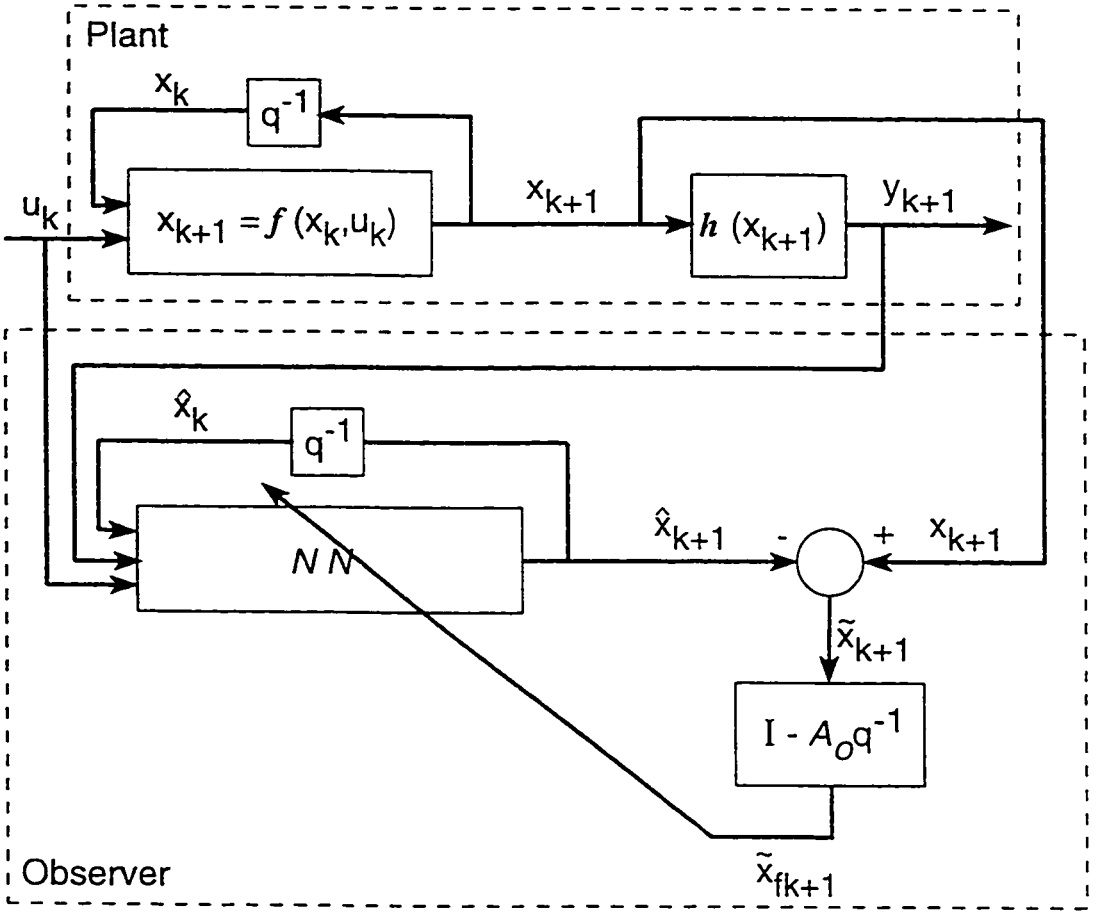


Figure 5.8: Dynamic state estimation scheme 5

update equation is given by Eq 5.54

$$J = \frac{1}{2}\sum_{k=1}^{n} \|\tilde{x}_{fk+1}\|^2 \tag{5.53}$$

$$\begin{aligned} W_{k+1} &= W_k + \eta \sum_{k=1}^{n} \Gamma_{k+1}\tilde{x}_{fk+1} \\ \text{where} \qquad \Gamma_{k+1} &= \left(\frac{\partial \hat{x}_{fk+1}}{\partial W}\right)^T \end{aligned} \tag{5.54}$$

$\frac{\partial \hat{x}_{fk+1}}{\partial W}$ can be calculated by using the following equations

$$\begin{aligned} \frac{\partial \hat{x}_{fk+1}}{\partial W} &= \frac{\partial \hat{x}_{k+1}}{\partial W} - A_0 \frac{\partial \hat{x}_k}{\partial W} \\ \frac{\partial \hat{x}_{k+1}}{\partial W} &= \frac{\partial^+ \hat{x}_{k+1}}{\partial \hat{x}_k}\frac{\partial \hat{x}_k}{\partial W} + \frac{\partial^+ \hat{x}_k}{\partial W} \end{aligned} \tag{5.55}$$

These equations are derived from the linearized diagram of scheme 5 as given in Fig 5.9. Block partial derivatives and Masons rule are applied to get these expressions.

5.2.6 Scheme 6

This scheme is basically the same as scheme 5 except that the observer is of reduced-order. The scheme is developed following the idea of reduced order observer for linear systems as given in section 4.2.4. If some of the states of the system are accessible as outputs of the system, then only those states need to be observed which are not available as outputs. The block diagram for this scheme is shown in Fig 5.10. The order reduction block passes only those states of the system which are to be observed.

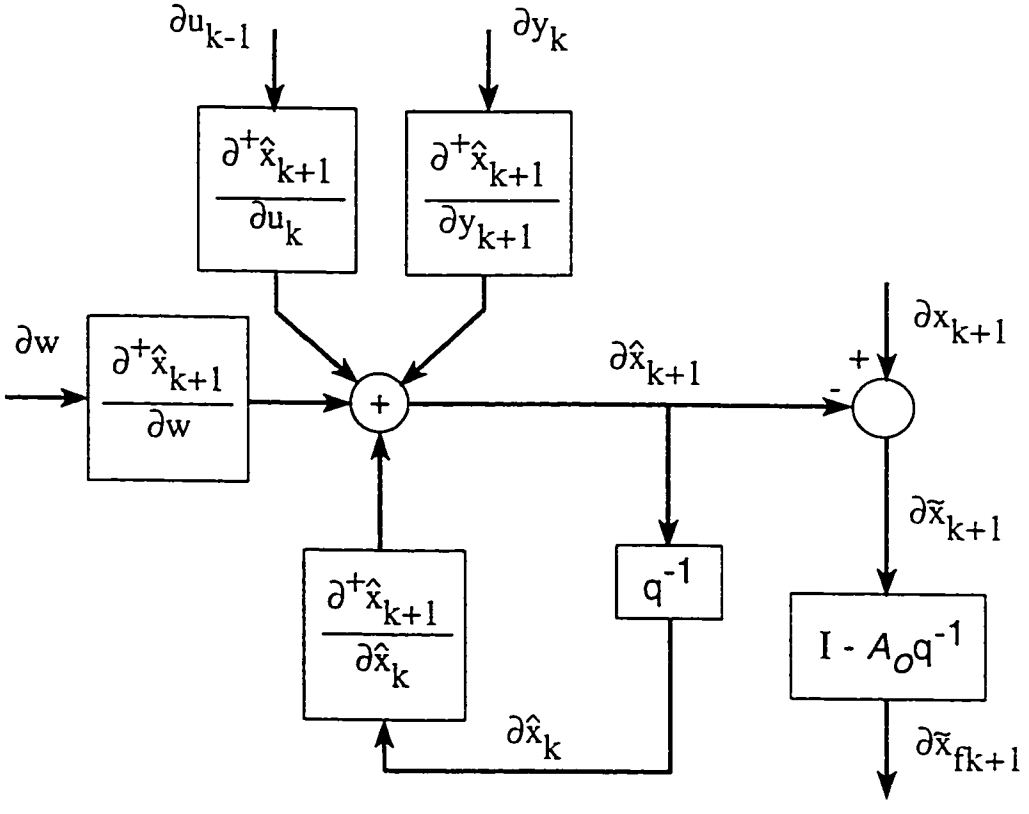


Figure 5.9: Linearized diagram for scheme 5

The observer equation for linear case (Eq 4.25) is extended to nonlinear case and is given by

$$\hat{z}_{k+1} = L(\hat{z}_k, y_{k+1}, u_k) \tag{5.56}$$

where L is a nonlinear function to be approximated by a neural network. Here we assume z_k to be a subset of x_k. The criterion function is given in Eq 5.57 and the weight update equation is given by Eq 5.58.

$$J = \frac{1}{2}\sum_{k=1}^{n} \|\tilde{z}_{fk}\|^2 \tag{5.57}$$

$$\begin{aligned} W_{k+1} &= W_k + \eta \sum_{k=1}^{n} \Gamma_{k+1}\tilde{z}_{fk} \\ \text{where} \qquad \Gamma_{k+1} &= \left(\frac{\partial \hat{z}_{fk}}{\partial W}\right)^T \end{aligned} \tag{5.58}$$

Where z are the states of the reduced order observer. $\frac{\partial \hat{z}_{fk}}{\partial W}$ can be calculated by using the following equations

$$\begin{aligned} \frac{\partial \hat{z}_{fk}}{\partial W} &= \frac{\partial \hat{z}_{k+1}}{\partial W} - A_0 \frac{\partial \hat{z}_k}{\partial W} \\ \frac{\partial \hat{z}_{k+1}}{\partial W} &= \frac{\partial^+ \hat{z}_{k+1}}{\partial \hat{z}_k}\frac{\partial \hat{z}_k}{\partial W} + \frac{\partial^+ \hat{z}_k}{\partial W} \end{aligned} \tag{5.59}$$

while $\tilde{z}_{fk+1}$ and $\tilde{z}_k$ are given by

$$\begin{aligned} \tilde{z}_{fk+1} &= \tilde{z}_{k+1} - A_0 \tilde{z}_k \\ \tilde{z}_k &= z_k - \hat{z}_k \end{aligned} \tag{5.60}$$

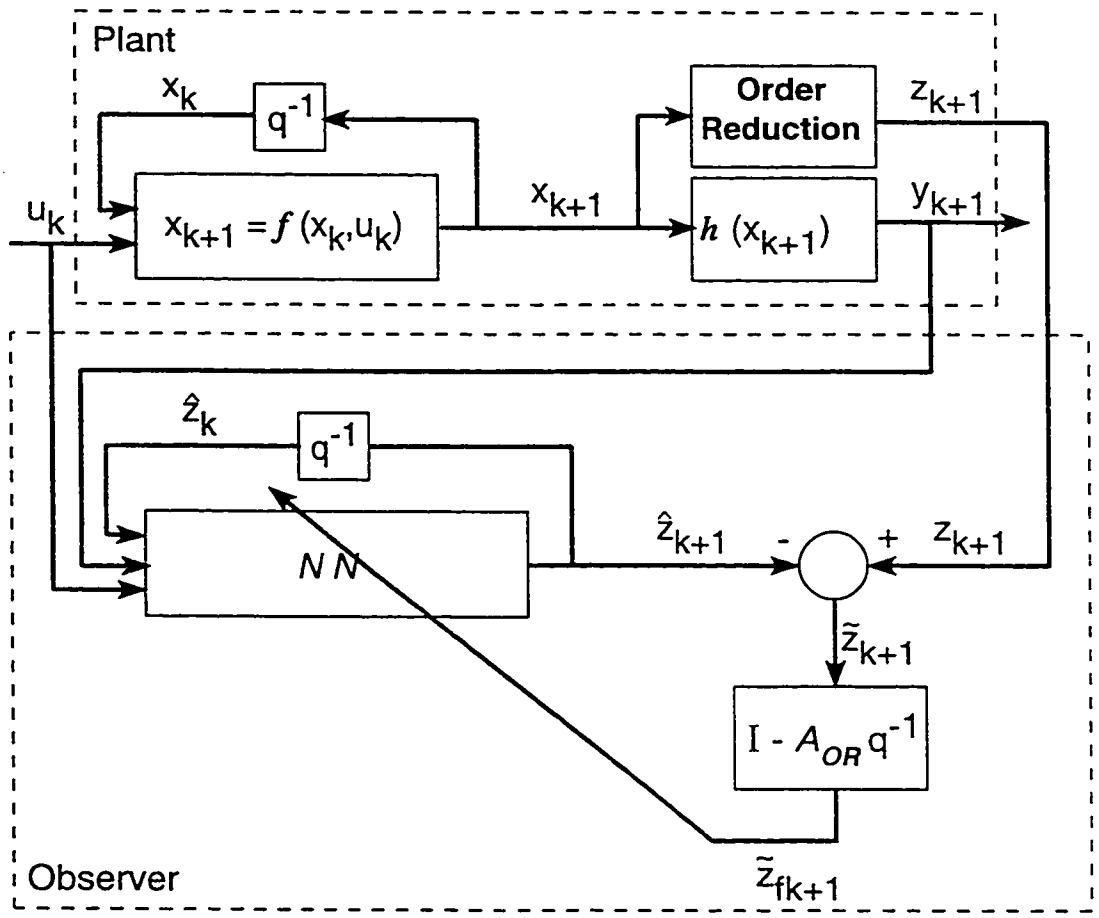


Figure 5.10: Dynamic state estimation scheme 6

Chapter 6

Stability and Adaptation Rate

In this chapter a stability analysis of the training scheme for the proposed neural observers is given.

The Neural observer design schemes are based on the training of the neural network through minimization of errors using a gradient descent algorithm. The training algorithms generally are not globally stable. However, the training of the neural networks is conducted off-line and in cases where the stability problem occurs or the algorithm stucks in a local minima, the training algorithm can be restarted with a different set of initial weights. Further, the adaptation rate can be altered to attain the global minimum or a satisfactory minimum point. A variation in the number of neuron layers and the number of neurons in each layer can also be done in obtaining a satisfactory observer. Thus the lack of global stability proof should not be taken as

a serious drawback of the scheme.

6.1 Stability of the Gradient Computation

In the following we consider the local stability of the gradient computation. The neural observer schemes consider a model of the system which along with the neural network is used for computing the weight gradients Γ_k. Assume that there exists an optimal weight W^* of the neural network that results in a locally stable dynamic neural observer for a given region of the operating point. Then, for a weight W of the neural network near the optimal weight W^*, the estimated states and the output of the observer respectively, will closely follow the actual states and the output of the system. This will result in a small and bounded value for the output error e_k.

Now, consider the system of Eq 5.1 and its linearized model given by Eq 5.4. The weight gradient equation for the neural observer of scheme 1 is given by Eq 5.25 which is reproduced below

$$\Gamma_{k+1} = \left(\frac{\partial \hat{y}_{fk+1}}{\partial W} \right)^T \tag{6.1}$$

Using equations Eq 5.26 and Eq 5.28 we can write the above sensitivity derivative as

$$\Gamma_{k+1}^T = FC_k(I - Hq^{-1})^{-1} e_k \frac{\partial^+ L_k}{\partial W} \tag{6.2}$$

Where H is given by Eq 5.29. Defining det and Adj respectively as the determinant

and the adjoint of a matrix, we get

$$\Gamma_{k+1}^T = \frac{FC_k \mathrm{Adj}(I - Hq^{-1})}{\det(I - Hq^{-1})} e_k \frac{\partial^+ L_k}{\partial W^*} \tag{6.3}$$

Using the linearized model of scheme 1 as given in Fig 5.3, the equation for the linearized configuration of the closed loop observer can be written as

$$\begin{aligned} \frac{\partial \hat{y}_{k+1}}{\partial u_k} &= C_k(I - Hq^{-1})^{-1} B_k \\ &= \frac{C_k \mathrm{Adj}(I - Hq^{-1})}{\det(I - Hq^{-1})} B_k \end{aligned} \tag{6.4}$$

Since the optimal observer is assumed to be stable, the locally linearized model will be stable too. Now it follows that the denominator polynomial of equation Eq 6.4 will be schur. It can be observed that Eq 6.3 and Eq 6.4 shares the same characteristic polynomial. This shows that the gradient computation as given in Eq 6.3 is also stable near the optimal weight W^*.

6.2 Adaptive Learning Rate

In this section an algorithm developed by Ahmed [53] for the adaptive learning rate is presented. This algorithm ensures the local stability of the weight update algorithm. The algorithm is based on the linearization of error in the weight space. The weight update algorithm minimizes the cost function given below,

$$J = \frac{1}{2} \sum_{k=1}^{n} \|\varepsilon_{k+1}(W_k)\|^2 \tag{6.5}$$

where $\varepsilon_{k+1}(W_k)$ is the difference between the actual and the desired output. The weight update equation can be written as follows.

$$W_{k+1} = W_k + \eta_k \sum_{k=1}^{n} \Gamma_{k+1}(W_k)\varepsilon_{k+1}(W_k) \tag{6.6}$$

Let W^* be the optimal weight matrix that minimizes the cost function of Eq 6.5. Then it must satisfy the optimality condition.

$$\sum_{k=1}^{n} \Gamma_{k+1}(W^*)\varepsilon_{k+1}(W^*) = 0 \tag{6.7}$$

Assuming that W_k lies in the vicinity of W^* and using the first order approximation of $\varepsilon_{k+1}(W_k)$, we have

$$\varepsilon_{k+1}(W_k) \approx \varepsilon_{k+1}(W^*) + \Gamma_{k+1}^T(W^*)\tilde{W}_k \tag{6.8}$$

and

$$\Gamma_{k+1}(W_k) \approx \Gamma_{k+1}(W^*) \tag{6.9}$$

where $\tilde{W}_k \equiv W^* - W_k$. Equation 6.6 can be written as

$$W_{k+1} \approx W_k + \eta_k \sum_{k=1}^{n} \Gamma_{k+1}(W_k)\Gamma_{k+1}^T(W_k)\tilde{W}_k + \eta_k \sum_{k=1}^{n} \Gamma_{k+1}(W^*)\varepsilon_{k+1}(W^*) \tag{6.10}$$

By using Eq 6.7, we can drop the last term. By subtracting both sides from W^* and pre-multiplying both sides by their transposes give

$$\|\tilde{W}_{k+1}\|^2 \approx (1 - 2\eta_k\alpha + \eta_k^2\alpha\beta)\|\tilde{W}_k\|^2 \tag{6.11}$$

where $0 \leq \lambda_{min}(G) \leq \alpha$.

$$\begin{aligned} \beta &\leq \lambda_{max}(G) \leq \gamma, \\ G &\equiv \sum_{k=1}^{n} \Gamma_{k+1}(W_k)\Gamma_{k+1}^T(W_k), \\ \text{and} \qquad \gamma &\equiv tr(G) = \sum_{k=1}^{n} \|\Gamma_{k+1}(W_k)\|^2 \end{aligned} \tag{6.12}$$

$\lambda_{min}(.)$, $\lambda_{max}(.)$, $tr(.)$ and $\|.\|^2$ are the maximum and minimum eigenvalues, trace and F-norm respectively. It has been verified in [55] that the expression for η_k

$$\eta_k = \frac{\mu_k}{\gamma + \epsilon}; \qquad 0 \leq \mu_k \leq 2 \tag{6.13}$$

ensures that

$$\|\tilde{W}_{k-1}\|^2 \leq \|\tilde{W}_k\|^2 \tag{6.14}$$

where ϵ is a small positive quantity introduced in order to avoid division by zero. In our simulations the value of ϵ is taken as 0.001. Equations Eq 6.13 and Eq 6.12 constitute the adaptation algorithm. The optimum value of μ_k is 1 but this value and the range given in Eq 6.13 can be ignored as the derivation is based on linearization approximation which are poor in the initial stages. In our simulations we used the following conditions to update the value for μ_k.

$$\mu_{k+1} = \begin{cases} 1.05\mu_k & \text{if} \quad J_k < 0.99 J_{k-1} \\ 0.7\mu_k & \text{if} \quad J_k > 1.02 J_{k-1} \\ \mu_k & \text{Otherwise} \end{cases} \tag{6.15}$$

Use of equation Eq 6.13 ensures the local stability of the weight update algorithm.

Chapter 7

Simulation

This chapter describes the simulation studies that were done for all the proposed schemes. A comparative study of the proposed schemes with the static observer and the extended Kalman filter is also included.

First, a feasibility study of all the proposed schemes is performed. Neural observers based on the proposed schemes are designed and tested on a system taken from Ref. [42]. Next, a comparative study among the proposed schemes alongwith the static observer and extended Kalman estimator is conducted. Robustness of the observers are studied in the presence of model variations and disturbances. Finally, the best among the proposed schemes is implemented on the design of observers for Van der Pol's equation and an inverted pendulum described by a nonlinear fourth order equation.

After the design of the observers, they have been tested on plant inputs that were different than the training inputs. For all the dynamic observers, the chosen initial estimate has been taken as different compared to the true ones. This practice was not followed for the static observers which does not require an initial state estimate. This goes in favour of the static scheme during the comparison studies.

The performance of the observers has also been tested by taking different initial conditions (i.e. $\hat{x}(0)$ and $x(0)$). The performance in general deteriorated with the increase in the difference, as expected. When this difference became too large, at times, they diverged. The range for which they converge depend on the training set.

The performance of the observer is evaluated for each state in terms of the sum of squared error (SSE) between the observed states and the actual states over a fixed simulation interval. An overall observer performance is also provided by adding SSE's for all states. SSE for each state is also presented separately in order to aid comprehensive comparison.

Multi-layered feed-forward (MFNN) neural networks with two hidden layers are considered for all the neural observer designs. The number of neurons in the input and the output layers of the network were fixed by the observer structure as shown in chapter 5. The number of neurons in the hidden layer were chosen based on trial and error. The training iterations have been carried on for atmost 1000 iterations or until the criterion function has reached 0.02. In the cases of static scheme and schemes 5

and 6, where the system information is only available through the inputs and outputs of the plant, and no predicted values of state are available, the maximum number of iterations were increased to 30,000.

7.1 Example 1

A nonlinear third order BIBO system given by the following equations is considered [42].

$$\begin{aligned} x_1(k+1) &= 0.5x_3(k) \\ x_2(k+1) &= x_1(k) + [1 - 0.4x_2(k)]u(k) \\ x_3(k+1) &= 0.3x_1(k)x_3(k) - x_2(k) \\ y(k) &= x_1(k) \end{aligned} \tag{7.1}$$

The linearized model around an operating point $\bar{x}(k)$, is given by

$$\begin{aligned} \hat{x}_1(k+1) &= 0.5\hat{x}_3(k) \\ \hat{x}_2(k+1) &= \hat{x}_1(k) + [1 - 0.4\bar{x}_2(k)]\hat{u}(k) - 0.4\bar{u}(k)\hat{x}_2(k) \\ \hat{x}_3(k+1) &= 0.3\hat{x}_1(k)\bar{x}_3(k) + 0.3\bar{x}_1(k)\hat{x}_3(k) - \hat{x}_2(k) \\ \hat{y}(k) &= \hat{x}_1(k) \end{aligned} \tag{7.2}$$

while the linearization at the origin is given by

$$\begin{aligned} \delta x(k+1) &= \begin{bmatrix} 0 & 0 & 0.5 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} \delta x_1 \\ \delta x_2 \\ \delta x_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} u(k) \\ \delta y(k) &= \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \delta x(k) \end{aligned} \tag{7.3}$$

Observability of the linearized system is checked at the origin and the observability matrix is found to be of full rank. Throughout the training of the neural observers for this example, the system is simulated by taking random initial conditions and applying a uniformly distributed random input $u_k \in [-2, 2]$. After the training is completed a simulation length of 50 time steps is considered for the validation and comparative studies of the observers.

7.1.1 Scheme 1

A neural observer based on scheme 1, described in the section 5.2.1 is implemented. The block diagram for the observer configuration is as shown in Fig 5.2. Structure of the neural network consisted of 4 and 3 neurons in the input and output layers respectively. The two hidden layers had 5 and 8 neurons respectively.

When the training is completed, the observer is validated by simulating the system and the observer configuration with same input but different initial conditions and then comparing the estimated states with the actual ones. Figure 7.1 shows the actual

states of the system alongwith the states obtained by the dynamic observer. The sum of the squared errors is also shown in the figure. From the figure it is evident that the error is mainly during the first few time steps due to different initial conditions.

7.1.2 Scheme 3

The dynamic neural observer of scheme 3 is implemented based on the configuration given in Fig 5.5. The neural network structure is kept the same as in scheme 1. After the training of the observer the validation test is performed by applying the same set of inputs as were applied in scheme 1. The initial conditions for the system and the observer are also kept in accordance with scheme 1. Figure 7.2 shows the actual and estimated states. It can be observed that after the first few time steps, the observed states follow the actual states almost exactly. The observer is tested for robustness which will be discussed later.

7.1.3 Static neural observer

Static neural observer scheme is also implemented and will be used for comparison purposes. First, the system of Eq 7.1 is simulated for 300 time steps and the data for the input, output and the states of the system is obtained. This data is used to train the neural network. The input, output and their delayed values form the input vector for the static observer, while the states of the system and the neural output are used

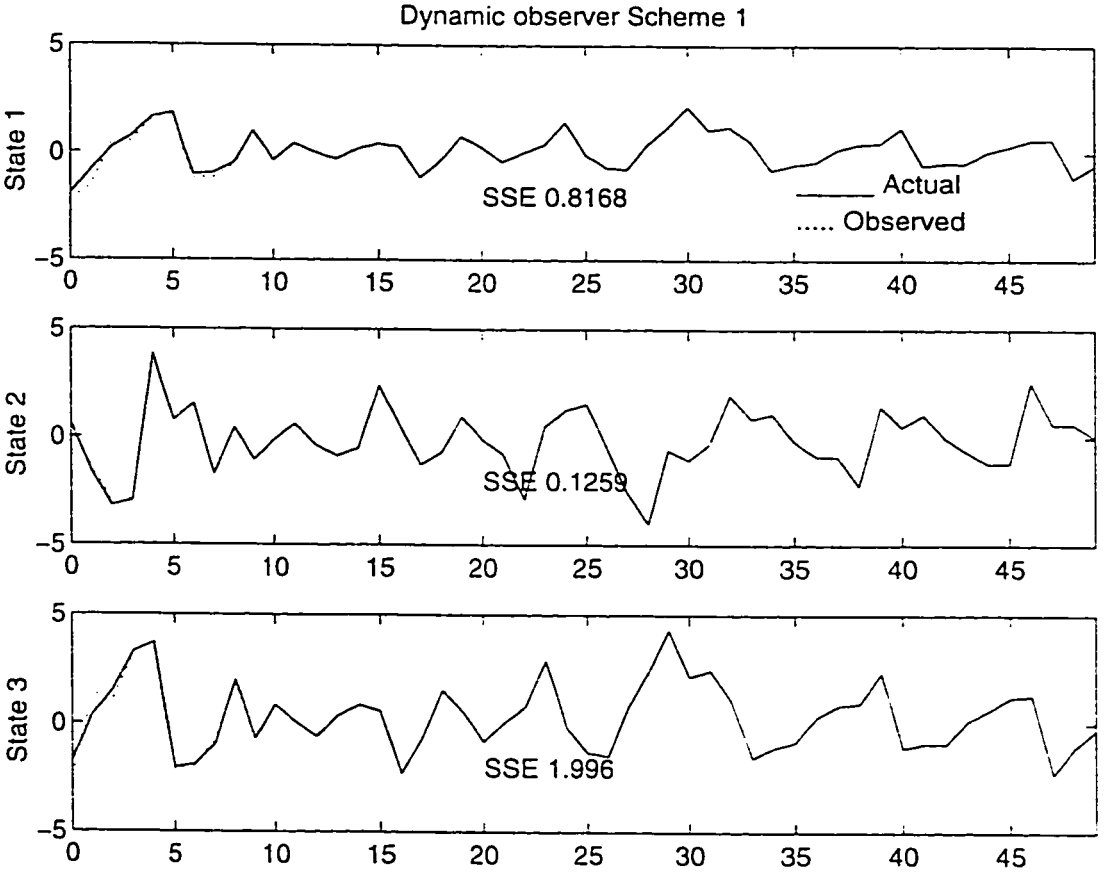


Figure 7.1: Estimated states by scheme 1

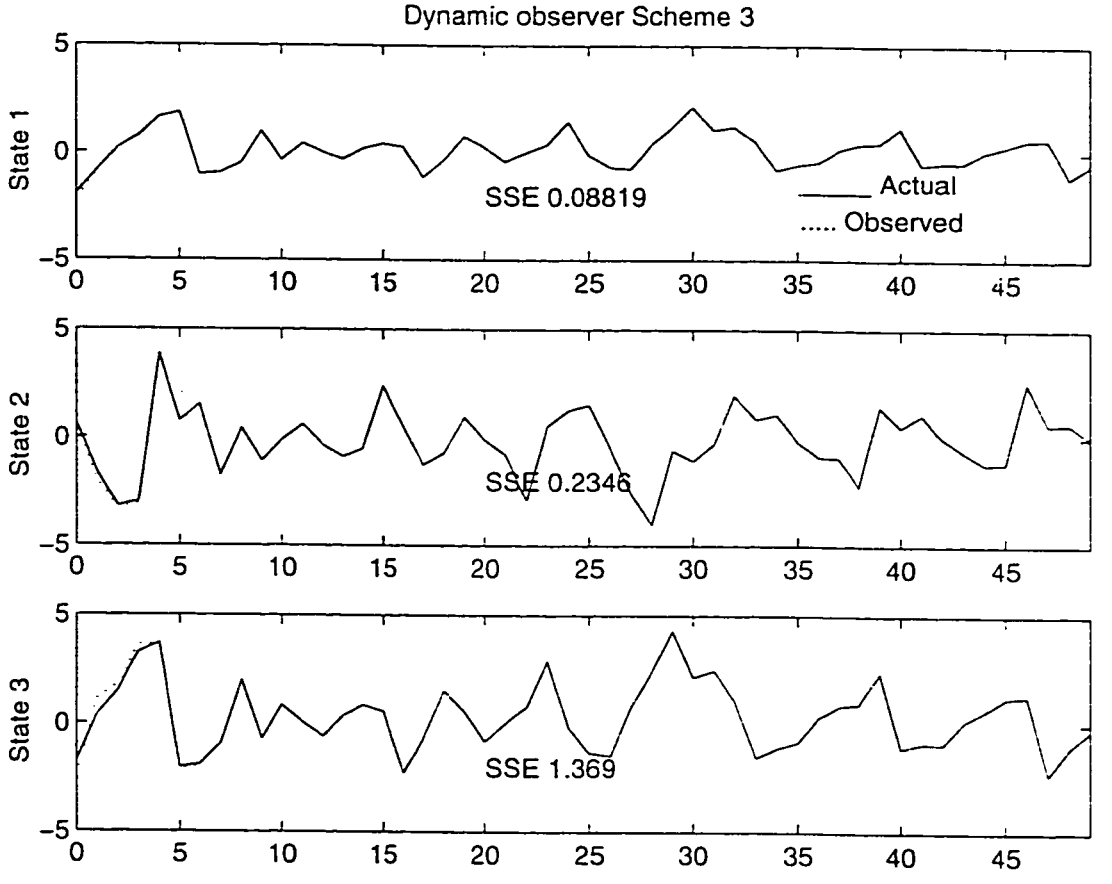


Figure 7.2: Estimated states by scheme 3

to calculate the criterion function. Training of the neural networks is done for 30,000 iterations. The structure of the neural network is taken as follows: 5 neurons in the input layer, 3 neurons in the output layer and two hidden layers of 12 and 6 neurons respectively. The training scheme is done as shown in Fig 5.1. Figure 7.3 shows the actual states of the plant and the estimated states as obtained by the static neural observer. Unlike the dynamic observer schemes 1 and 3, some discrepancies in the actual and the estimated states can be seen in the later stages of the simulation. It can further be observed that the SSE's are larger for the static observer.

7.1.4 Extended Kalman filter

Extended Kalman filter algorithm is also implemented and the results for the Kalman estimates and the actual states are shown in Fig 7.4. The estimates are found to better than neural observers in terms of the SSE's but the computations involved in the linearization of the system and the Kalman gain calculations during the estimation process are large. Similar inputs and initial conditions are applied to all the above schemes with the exception of static observer which does not require any initial condition. A comparison of the results of the four schemes shows that EKF performs better in terms of sum of the squared error. Where as the results of the dynamic schemes are better compared to the static observer. The advantage of the dynamic schemes over EKF is the reduced computations at the implementation stage of the observer.

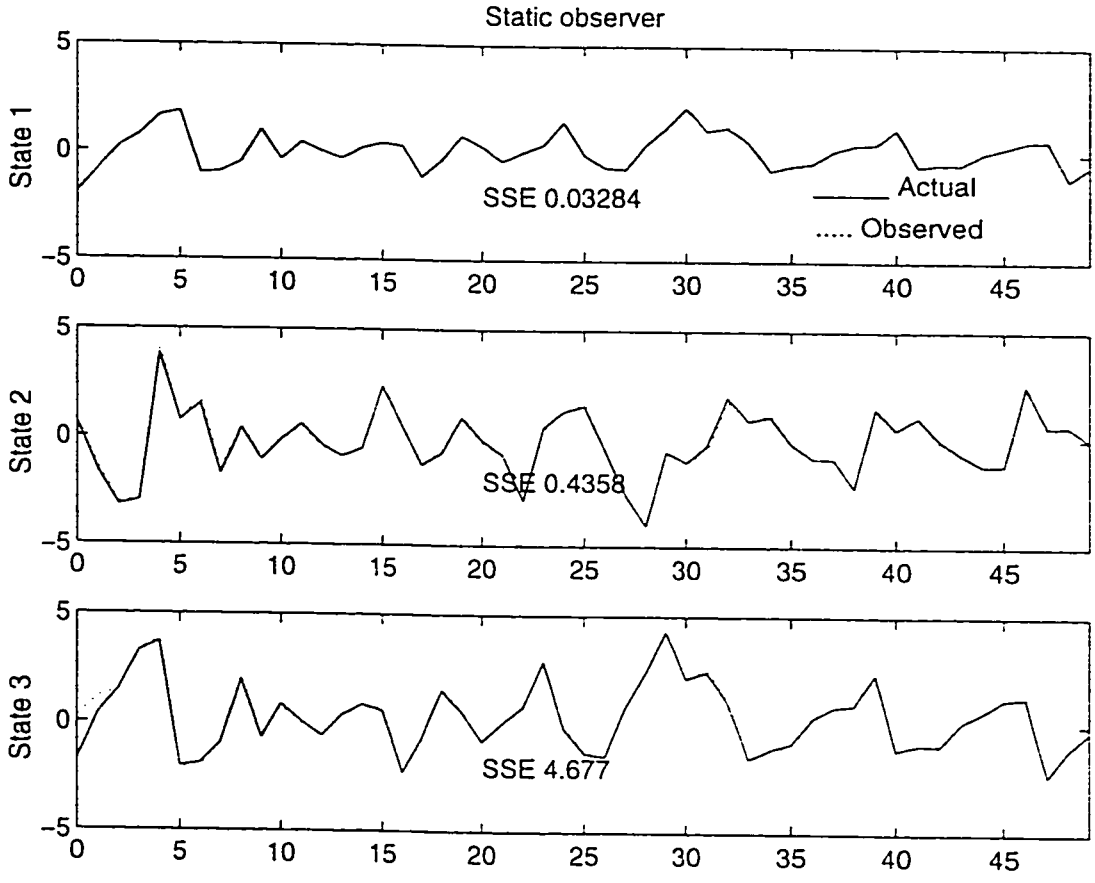


Figure 7.3: Estimated states by static neural observer

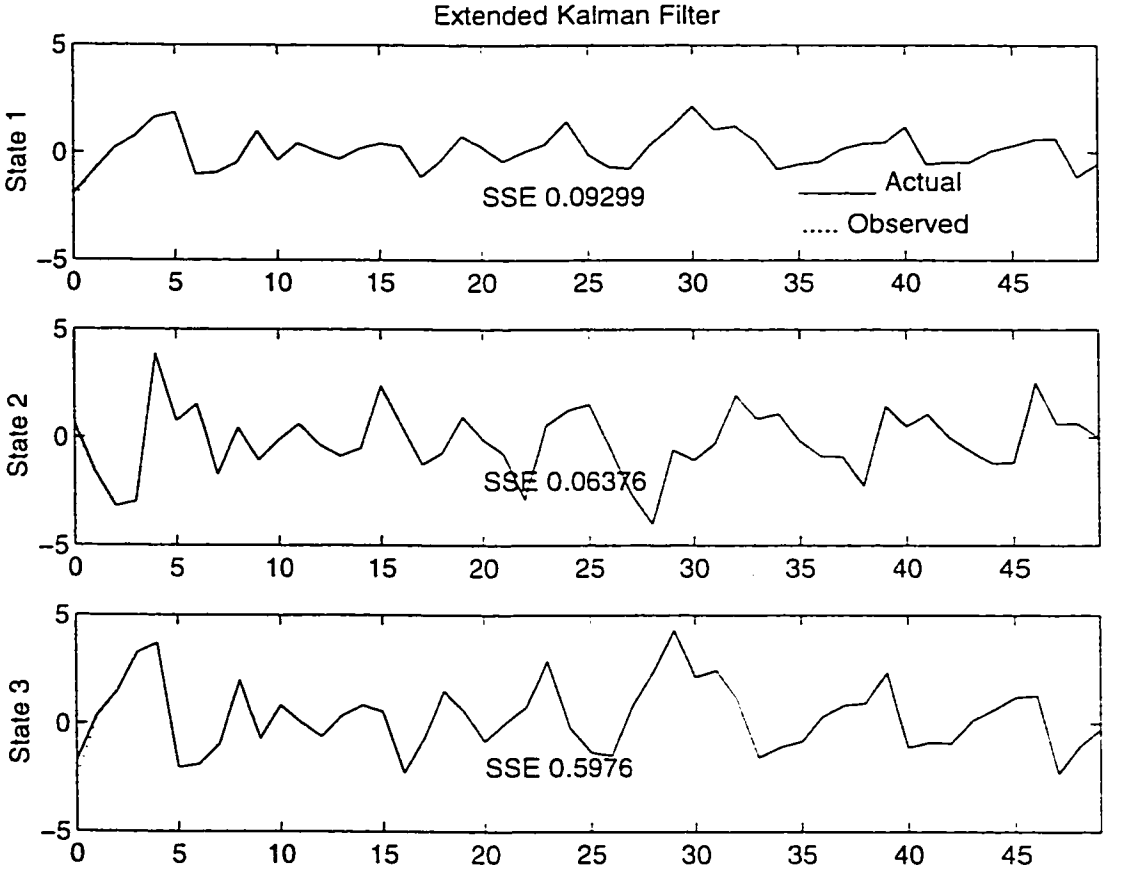


Figure 7.4: Estimated states by extended Kalman filter

The observer of scheme 3 performs better than scheme 1 of dynamic observers as in the correction stage, the former uses more recent data.

7.1.5 Comparative study of all schemes

Schemes 2, 4 and 5 of the dynamic neural observers are also applied on the system of Eq 7.1. The training schemes were implemented as developed in chapter 5. After the training is completed a rigorous test of all the observers is conducted.

For a comparative study, all the observers were initialized by the same initial conditions (except the static observer as discussed earlier). The system was initialized by a different set of initial conditions. Both the observer and system initial conditions are taken different from the ones used during the design stage of the neural observers. Same inputs are applied to the system and to all the observer configurations and the sums of the squared errors is calculated for each estimated state as well as for all the states. These values are given in Table 7.1. It can be observer that the sum-squared error obtained for the dynamic observers are small and that they outperform the static scheme. Scheme 3 came out to be the best one among all the neural observers. EKF has the minimum sum-squared errors but that is at a considerable computational cost. The EKF, for this example, requires about 120 multiplication operations and 82 addition operations in addition to the computations involved in linearizing the system. These computations depend on the complexity and the nonlinearity of the

Table 7.1: Sum-squared error when random input is applied

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	2.9388	1.8946	1.6920	2.8392	3.0710	0.7469	5.1452
State 1	0.8168	0.4355	0.0882	0.0930	0.2840	0.0928	0.0328
State 2	0.1259	0.1747	0.2346	0.3652	0.8116	0.0637	0.4358
State 3	1.9961	1.2844	1.3692	2.3810	1.9754	0.5904	4.6766

Table 7.2: Structure of neural observers

Layer	Sch. 1	Sch. 2	Sch. 3	Sch. 4	Sch. 5	Sch. 6	Static
Input	4	4	4	4	5	4	5
1^{st} Hidden	5	5	5	5	12	12	12
2^{nd} Hidden	8	8	8	8	6	6	6
Output	3	3	3	3	3	2	3

system. For this example 4 multiplication and 2 addition operations are required during the linearization, but for the inverted pendulum as given in example 3, about 120 multiplication and 25 addition operations will be needed. While for the neural observers the computations involved depend on the neural network structure. During the implementation stage of schemes 1 to 4, a total of 100 multiplications and 85 addition operations with 13 exponential operations are required. But these operations can be avoided by using the commercially available neural chips.

The neural network structure chosen to design the neural observers for the given system are shown in Table 7.2.

The training of the neural observers was done on a Pentium PC (166MHz). The time taken during the design stage of these observers is given in Table 7.3. It can be observer from Table 7.2 and Table 7.3 that the computation time increases with the

Table 7.3: Computation time (minutes/hundred iterations)

Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	Scheme 6	Static
6.86	6.90	7.33	7.38	11.2	10.1	12.3

neural network size. The total computation time can be obtained by realizing that total iterations required by schemes 1-4 were in the range of 1000, while by schemes 5-6 and the static observer were in the range of 30,000.

7.1.6 Model variations

Simulation study has been conducted to check the robustness of the observers for model variations. The parameters of the system are varied and the performance of all the observer schemes in terms of sum-squared errors is calculated.

Introducing modelling errors in the system of Eq 7.1, the system equations can be written as

$$\begin{aligned} x_1(k+1) &= (0.6 + \Delta p_1)x_3(k) \\ x_2(k+1) &= (1 + \Delta p_2)x_1(k) + [(1 + \Delta p_3) + (-0.4 + \Delta p_4)x_2(k)]u(k) \\ x_3(k+1) &= (0.3 + \Delta p_5)x_1(k)x_3(k) - (1 + \Delta p_6)x_2(k) \\ y(k) &= x_1(k) \end{aligned} \tag{7.4}$$

One of the parameters is changed at a time, and the system with the observer configuration is simulated for a length of 50 time steps. The sum-squared errors for

Table 7.4: Sum-squared error when $\Delta p_1 = 0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	5.2901	5.2919	4.3473	5.8201	5.1045	1.5950	7.8715
State 1	2.4902	2.3528	0.1085	0.1038	0.5627	0.3212	0.1171
State 2	0.1406	0.5438	0.6724	0.5884	1.1549	0.1096	1.0741
State 3	2.6594	2.3953	3.5664	5.1279	3.3869	1.1642	6.6803

Table 7.5: Sum-squared error when $\Delta p_1 = -0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	1.6340	2.3531	2.0914	2.1490	4.0692	0.5703	5.5807
State 1	1.2691	1.5295	0.0296	0.0077	0.1381	0.1905	0.0219
State 2	0.0270	0.2255	0.3469	0.2521	1.8844	0.0354	0.3688
State 3	0.3379	0.5981	1.7149	1.8892	2.0467	0.3445	5.1900

each state as well as all states are calculated. Tables 7.4 to 7.11 give these measures for certain parameter variations. As can be seen in all the cases that the maximum deviation is in the third state of the system. Scheme 3 and in some cases scheme 4 perform the best among all the neural observers. EKF yielded a better response in all the cases considered here, but at an increased computational cost.

Table 7.6: Sum-squared error when $\Delta p_2 = 0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	1.5407	1.0703	1.3259	0.9277	2.3703	0.0298	6.1473
State 1	0.3248	0.1426	0.0052	0.0042	0.2040	0.0037	0.0379
State 2	0.4919	0.4263	0.4150	0.4608	0.8663	0.0025	1.1080
State 3	0.7240	0.5013	0.9057	0.4628	1.3001	0.0236	5.0014

Table 7.7: Sum-squared error when $\Delta p_2 = -0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	1.2627	1.1361	1.3878	0.8994	3.1146	0.0298	5.8354
State 1	0.2974	0.1574	0.0056	0.0040	0.1626	0.0037	0.0298
State 2	0.4166	0.4700	0.4653	0.5000	1.2969	0.0025	0.7859
State 3	0.5487	0.5087	0.9169	0.3955	1.6551	0.0236	5.0197

Table 7.8: Sum-squared error when $\Delta p_5 = 0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	5.1752	4.2145	5.0061	1.3877	4.7752	0.0298	6.4750
State 1	1.2805	0.9157	0.0121	0.0052	0.2088	0.0037	0.0360
State 2	0.0406	0.0900	0.5956	0.0571	0.5304	0.0025	0.6746
State 3	3.8541	3.2088	4.3983	1.3254	4.0390	0.0236	5.7644

Table 7.9: Sum-squared error when $\Delta p_5 = -0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	3.6182	2.5255	2.5317	1.5152	4.4568	0.0298	7.0987
State 1	1.1115	0.5534	0.0098	0.0071	0.1711	0.0037	0.0312
State 2	0.0208	0.0420	0.2367	0.0546	0.9358	0.0025	0.3493
State 3	2.4859	1.9301	2.2852	1.4535	3.3499	0.0236	6.7181

Table 7.10: Sum-squared error when $\Delta p_6 = 0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	1.6821	2.0490	1.9486	1.0524	3.8697	0.0298	5.9782
State 1	0.3975	0.4065	0.0102	0.0048	0.1407	0.0037	0.0254
State 2	0.0105	0.0713	0.1079	0.0700	0.8485	0.0025	0.3585
State 3	1.2740	1.5712	1.8304	0.9776	2.8806	0.0236	5.5942

Table 7.11: Sum-squared error when $\Delta p_6 = -0.1$

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	2.2207	2.4305	2.0331	1.3647	3.6602	0.0298	6.1429
State 1	0.4957	0.4824	0.0089	0.0062	0.2718	0.0037	0.0521
State 2	0.0119	0.0931	0.1221	0.0898	0.5438	0.0025	0.6408
State 3	1.7132	1.8549	1.9021	1.2688	2.8447	0.0236	5.4500

7.1.7 Effect of disturbances

Further testing of the observers are done by adding state and output noises in the system and performing the state estimation through the different observers. The state equations are now represented by:

$$\begin{aligned} x_1(k+1) &= 0.5x_3(k) + w_1(k) \\ x_2(k+1) &= x_1(k) + [1 - 0.4x_2(k)]u(k) + w_2(k) \\ x_3(k+1) &= 0.3x_1(k)x_3(k) - x_2(k) + w_3(k) \\ y(k) &= x_1(k) + v(k) \end{aligned} \tag{7.5}$$

where $w_i(k)$ and $v(k)$ respectively denote the state and measurement noise. The noise has been zero mean and Gaussian with variances of 0.1. State noise and output noise are added separately and sum of the squared errors of the observers are computed. Then both state and output noises are added simultaneously and the estimated states are observed. The results are shown in Tables 7.12 to 7.14.

The results show that the static observer scheme performs reasonably for estimation of state 1 and 2 but in the case of the third state the error is quite large thus

affecting the overall observer performance. Where as for the dynamic observers, the performance of scheme 3 is satisfactory under all conditions.

Table 7.12: Sum-squared error when state noise is added

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	2.1188	1.2160	2.0552	1.2470	3.6400	0.0518	21.2454
State 1	1.0073	0.4072	0.0171	0.0125	0.1136	0.0260	0.0382
State 2	0.2027	0.2120	0.5554	0.4365	1.6325	0.0108	4.2822
State 3	0.9087	0.5969	1.4827	0.7979	1.8939	0.0150	16.9250

Table 7.13: Sum-squared error when output noise is added

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	1.7941	0.7909	1.7653	1.7004	3.8223	0.0514	21.2997
State 1	0.8560	0.1228	0.2212	0.1859	0.3164	0.0258	0.2222
State 2	0.3363	0.2759	0.3753	0.5312	1.6202	0.0107	4.3651
State 3	0.6018	0.3922	1.1689	0.9833	1.8858	0.0150	16.7124

Table 7.14: Sum-squared error when state and output noise are added simultaneously

	Scheme 1	Scheme 2	Scheme 3	Scheme 4	Scheme 5	EKF	Static
All States	5.1085	1.7906	2.6279	2.3798	3.1334	0.1122	18.0943
State 1	2.7449	0.4758	0.1936	0.1681	0.2759	0.0252	0.1288
State 2	0.4582	0.4399	0.7358	0.7939	1.3326	0.0292	3.8786
State 3	1.9053	0.8749	1.6985	1.4178	1.5250	0.0577	14.0868

7.1.8 Scheme 6

A reduced order observer for the system of Eq 7.1 is also developed. The observer configuration is as shown in the Fig 5.10. Since state 1 is the output for this system,

Table 7.15: Sum-squared error for scheme 6

	Scheme 6	EKF	Static
All States	14.9782	0.0251	11.6389
State 2	6.3607	0.0248	6.5840
State 3	8.6175	0.0003	5.0196

reduced order observer is designed to estimate states 2 and 3. After the training, the reduced order observer is compared with the estimates of the Kalman filter and static neural observer. The sum of the squared errors is given in Table 7.15. The performance of this scheme is found to deteriorate in the presence of noise and uncertainties in the system.

7.2 Example 2: Van der Pol's Equation

Van der Pol's equation is a second order nonlinear differential equation. A form of the Van der Pol's equation can be given as [9]

$$\ddot{y} - \mu(1 - y^2)\dot{y} + 9y = w(t) \tag{7.6}$$

The state space representation of the Van der Pol's equation can be written as

$$\begin{aligned} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} &= \begin{bmatrix} x_2 \\ -9x_1 + \mu(1 - x_1^2)x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w(t) \\ y(t) &= x_1(t) + v(t) \end{aligned} \tag{7.7}$$

Table 7.16: SSE for Van der Pol's equation

Σ_w	Σ_v	Dynamic	EKF	Static
0.0	0.0	7.05	7.43	166.1
0.01	0.0	7.08	7.51	553.7
0.0	0.01	15.24	7.48	64284.1
0.01	0.01	15.48	7.58	64359.8
0.1	0.01	16.01	8.35	65004.6
0.01	0.1	94.74	8.24	124073.5
0.1	0.1	95.70	9.10	123932.0
1.0	0.1	100.89	16.77	124500.5
0.1	1.0	1430.11	16.12	154832.4
1.0	1.0	1438.15	24.43	153995.5

where $w(t)$ is the Gaussian wideband input and $v(t)$ is the measurement noise with variances Σ_w and Σ_v respectively. Many other nonlinear systems like the mass, spring damper system or equivalently an electrical RLC circuit with a nonlinear resistor can be converted into this form. Dynamic neural observer of scheme3 along with static neural observer and extended Kalman filter have been designed to estimate the states of the Van der Pol's equation. The value of μ is taken to be 0.5 and the sampling interval h is taken to be 0.05 sec. Fourth order Runge-Kutta's method is used to solve the equation. After the training of the neural observers, the observers are tested through simulation. The initial conditions for x(0) is taken as $(5, 0)$ while for $\hat{x}$ the initial value is taken as $(6, 2)$. The actual and the observed states are shown in Fig 7.5 and Fig 7.6 for the case when no noise is considered.

Table 7.16 shows the sum of the squared errors when different variances of noise are considered. It is evident from the table that the dynamic neural observer yielded

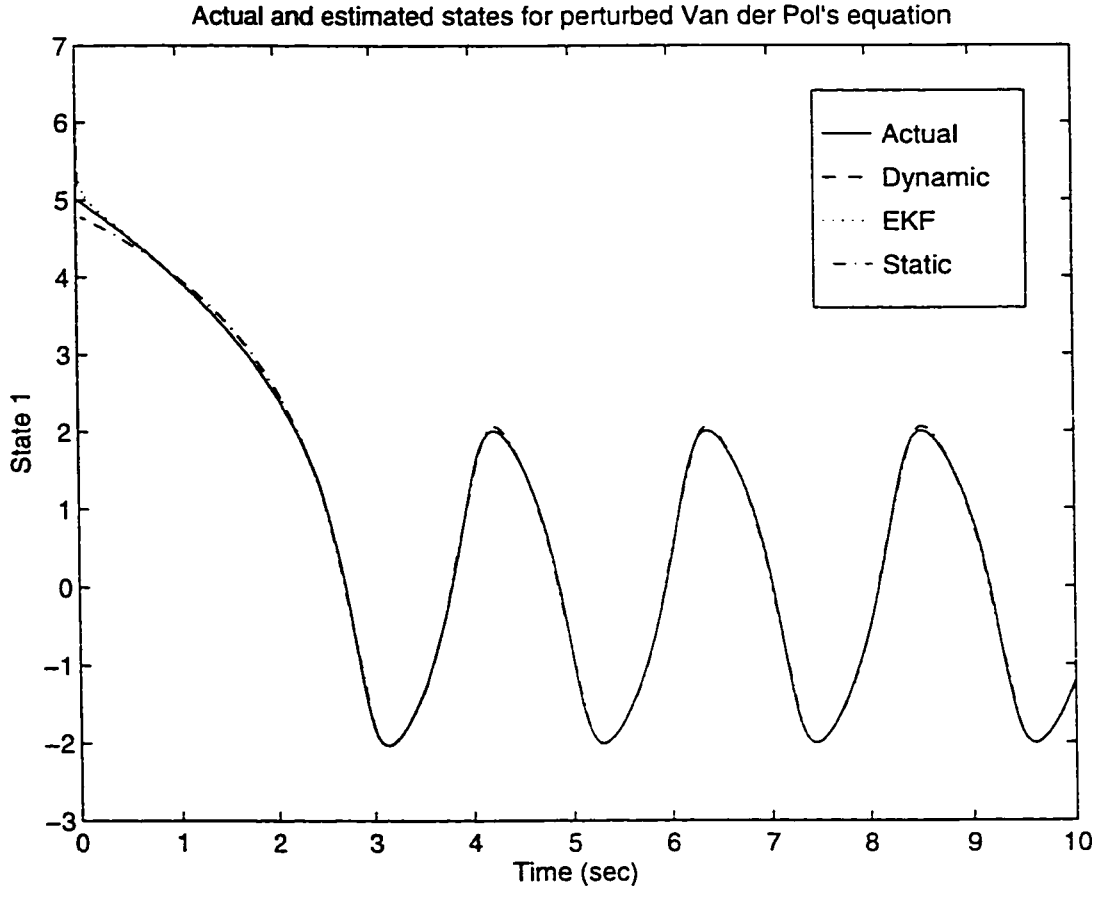


Figure 7.5: Estimated state 1 in Van der Pol's equation

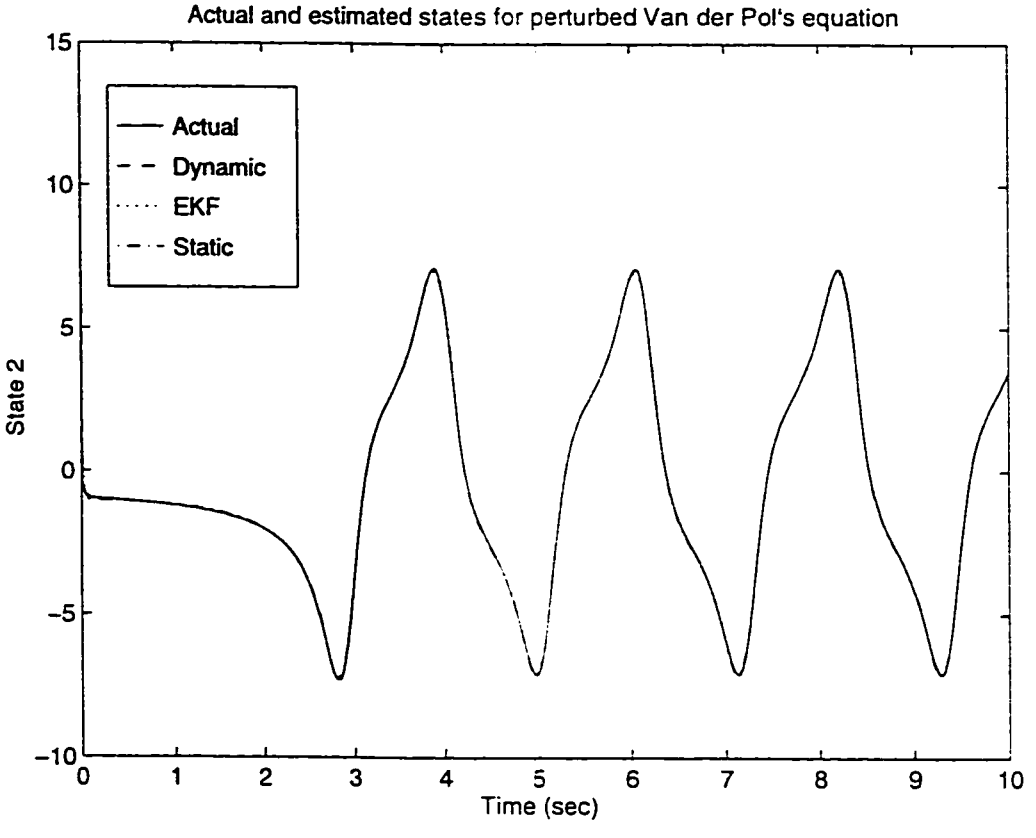


Figure 7.6: Estimated state 2 in Van der Pol's equation

Table 7.17: SSE for the perturbed Van der Pol's equation

Σ_w	Σ_v	Dynamic	EKF	Static
0.0	0.0	857.11	11545.6	130.23
0.01	0.0	857.58	11558.4	140.7
0.0	0.01	861.75	11763.2	1011.8
0.01	0.01	862.25	11775.6	1014.9
0.1	0.01	863.53	11802.5	1097.6
0.01	0.1	900.00	12244.8	6444.3
0.1	0.1	901.26	12269.7	6426.7
1.0	0.1	906.90	12347.9	6648.1
0.1	1.0	1374.7	13512.4	16880.9
1.0	1.0	1381.0	13553.5	17036.7

better performance than the static one. The EKF performed the best however it requires more computations.

7.2.1 Modelling variations

Next, modelling error is considered in the simulation. The value of μ for the simulation is taken as 2 while for observer design it is taken as 0.5. h is taken as 0.05. The estimated states after the training of the networks and the estimates by Kalman filter are shown in Fig 7.7 and Fig 7.8. Σ_w and Σ_v both have been been 0.01. The Kalman filter failed to track the states due to the modelling error.

Table 7.17 depicts the sum of the squared error for the three schemes for different noise variances. Static observer performed well in the absence of noise but a slight addition of noise deteriorated its performance considerably.

The actual and the estimated states for the dynamic and static neural observers,

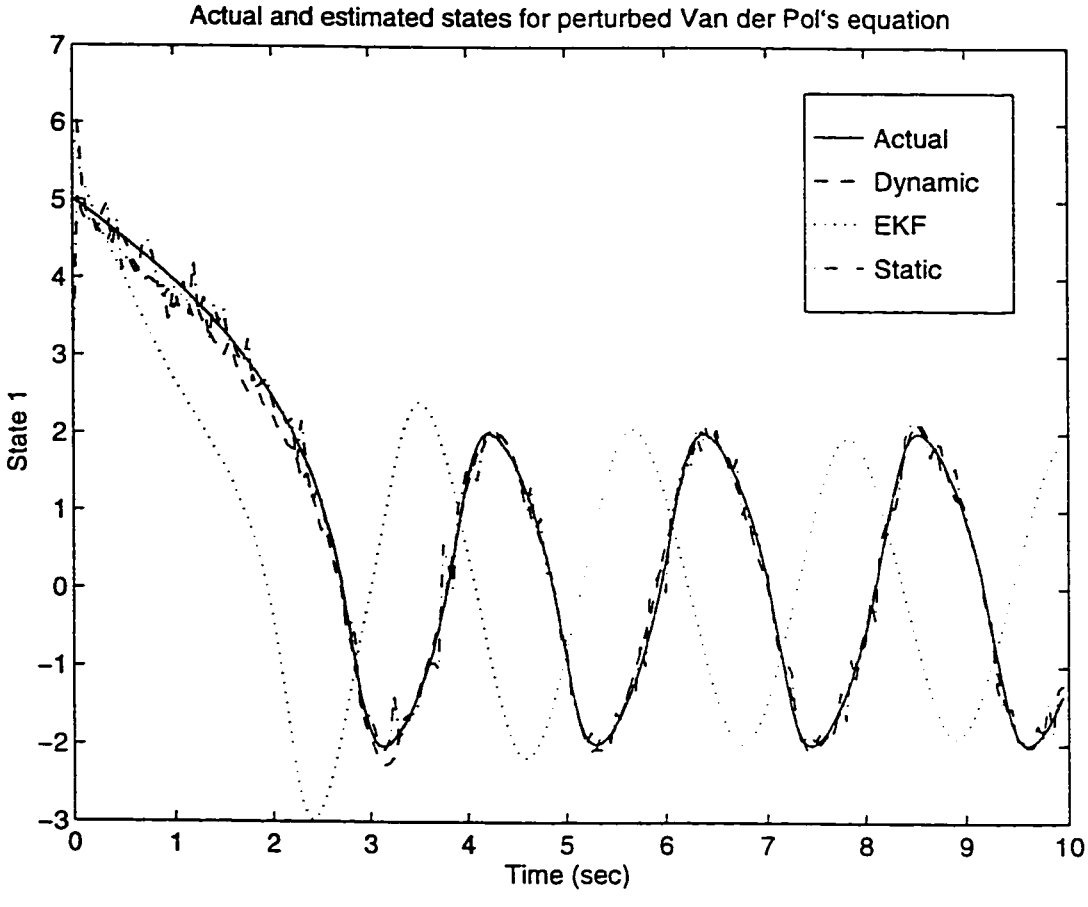


Figure 7.7: Estimated state 1 for the perturbed Van der Pol's equation

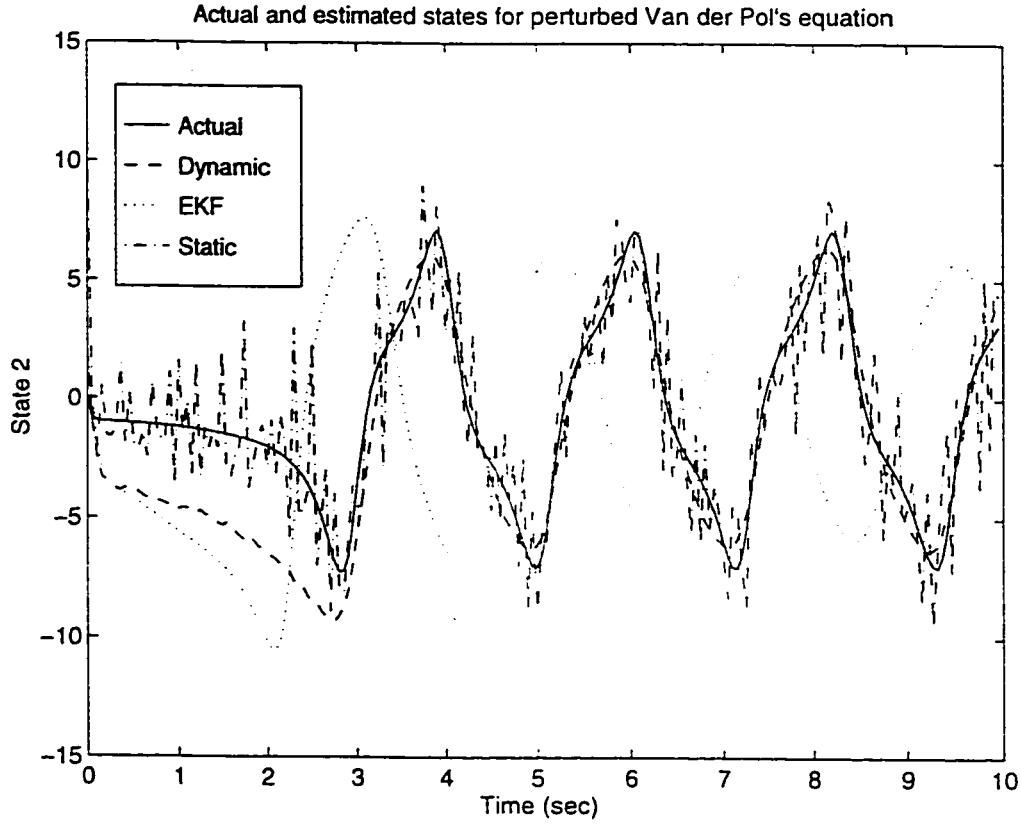


Figure 7.8: Estimated state 2 for the perturbed Van der Pol's equation

when $\mu = 0.5$ and $\Sigma_w = 0.1$ and $\Sigma_v = 0.01$ are shown in figures Fig 7.9 and Fig 7.10. It is evident the estimates by the dynamic observer are smooth while the estimates of the static observer are very bumpy.

7.3 Example 3: Inverted Pendulum

In this study a classical example of a single input multi-output inverted pendulum described by a fourth order nonlinear equation is considered. Let d be the distance of the cart from the centre and α be the angle of the pendulum. Defining the state vector x as $x = [d\ \dot{d}\ \alpha\ \dot{\alpha}]$ the system equations can be written as follows:

$$\begin{aligned}
\dot{x_1} &= x_2 \\
\dot{x_2} &= \frac{lx_3^2 \sin x_3 - g \sin x_3 \cos x_3 + \frac{1}{m}u}{\frac{M}{m} + \sin^2 x_3} \\
\dot{x_3} &= x_4 \\
\dot{x_4} &= \frac{\frac{M+m}{m} g \sin x_3 - lx_3^2 \sin x_3 \cos x_3 - \frac{\cos x_3}{m} u}{l\left(\frac{M}{m} + \sin^2 x_3\right)}
\end{aligned} \tag{7.8}$$

And the output equation is given by

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} x \tag{7.9}$$

A typical set of system parameters used for our design is given as follows:

Mass of the cart $M = 0.455Kg$,

Mass of the pendulum $m = 0.210Kg$,

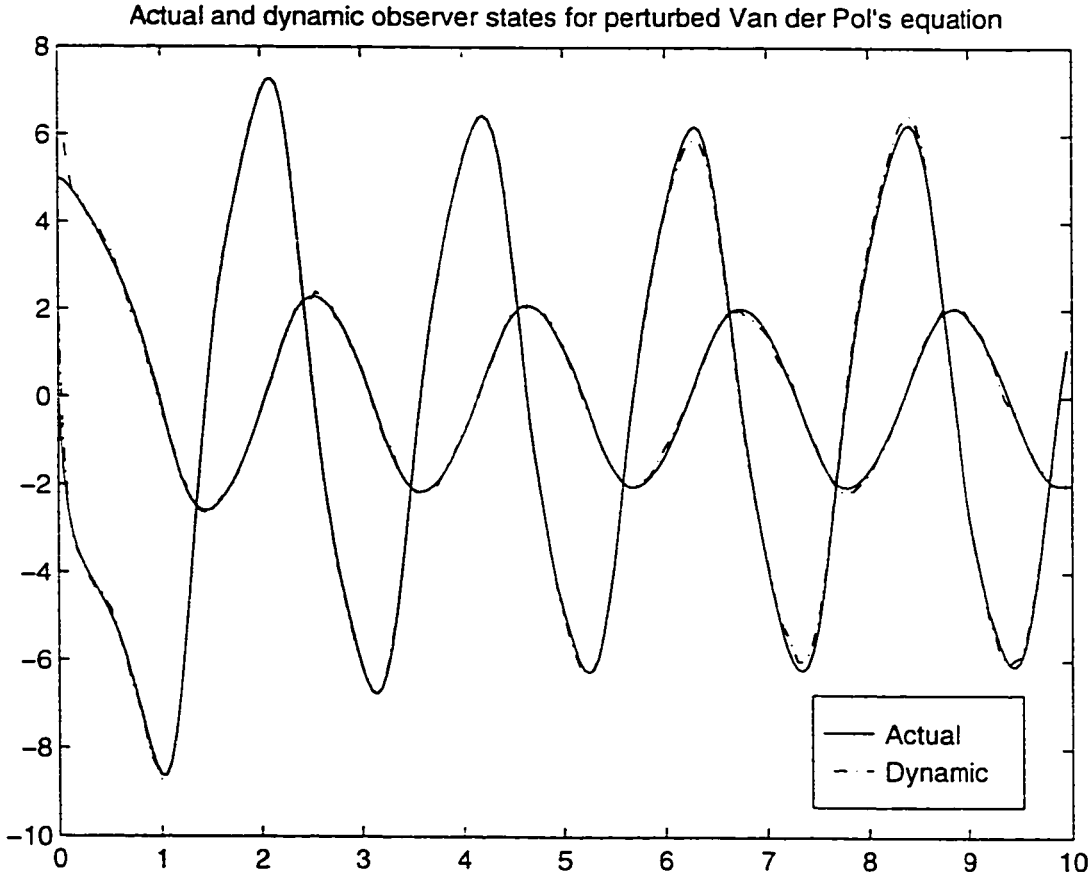


Figure 7.9: Dynamic observer estimates for the perturbed Van der Pol's equation

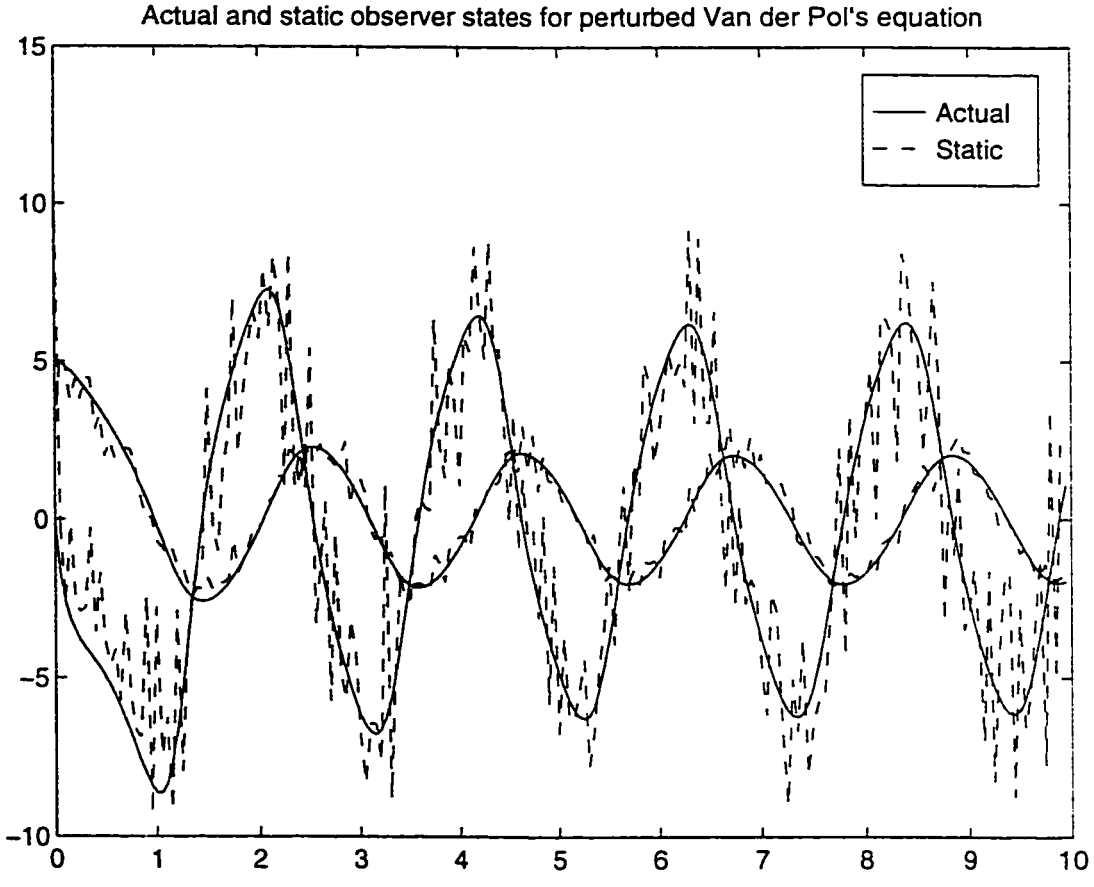


Figure 7.10: Static observer estimates for the perturbed Van der Pol's equation

Half length of pendulum rod $l = 0.305m$.

Acceleration due to gravity $g = 9.81m/sec^2$.

Input force applied to the cart, u in $Newtons$

As the cart is driven by a dc motor, the input force u is given by

$$\frac{K_m K_g}{R_m r_g} V - \frac{1}{R_m} \frac{K_m K_g^{\ 2}}{r_g} x_2 \tag{7.10}$$

where, V is the input force in $Volts$

K_m is motor torque constant $= 0.00767N - m/amp$

K_g is the gear ratio of the gear box of the dc motor $= 3.7$

R_m is the armature resistance $= 2.6ohms$

r_g is the radius of the output gear $= 0.0127/2m$

A stabilizing LQR controller is designed for the inverted pendulum by minimizing the cost function

$$J = \int (x^T Q x + r V^2) dt \tag{7.11}$$

where Q and r are positive definite and $Q \in \Re^{nxn}$. Considering

$$Q = diag \begin{bmatrix} 0.25 & 0 & 4 & 0 \end{bmatrix} \qquad \text{and} \qquad r = 0.0003 \tag{7.12}$$

Matlab control toolbox subroutine 'lqr' is used to obtain the gain matrix which is fine tuned on simulated the inverted pendulum and the final gain matrix is obtained as

$$K = \begin{bmatrix} -40 & -60 & -137.5 & -34.38 \end{bmatrix} \tag{7.13}$$

Table 7.18: SSE for the inverted pendulum

Σ_w	Σ_v	Dynamic	EKF	Static
0.0	0.0	16.5399	85.7783	276.3967
0.01	0.0	193.5662	125.5516	624.4574
0.0	0.01	17.7693	86.6363	445.3431
0.01	0.01	195.1424	126.6063	791.0748

The above gain matrix is used for all the design purposes.

Observer designs considered have been scheme 3 of dynamic neural observer, extended Kalman filter and the static neural observer.

After the training of the observers, the system and the observer are initialized with different initial conditions and simulated together in order to validate the schemes. The system is disturbed at time intervals 100 and 200 by changing the angle of the pendulum. The results are shown in the Figures 7.11 to 7.14 and the sum-squared errors are given in the Table 7.18

7.3.1 External disturbances

The error in the neural observers are found mainly in the derivative states of the inverted pendulum. The dynamic observer performed best in the absence of the noises and is superior to static observer in the presence of state and output noises. EKF on the other hand has a comparable performance in the presence of noise as shown in Table 7.18.

The performance of EKF improves when no disturbances in the form of forced

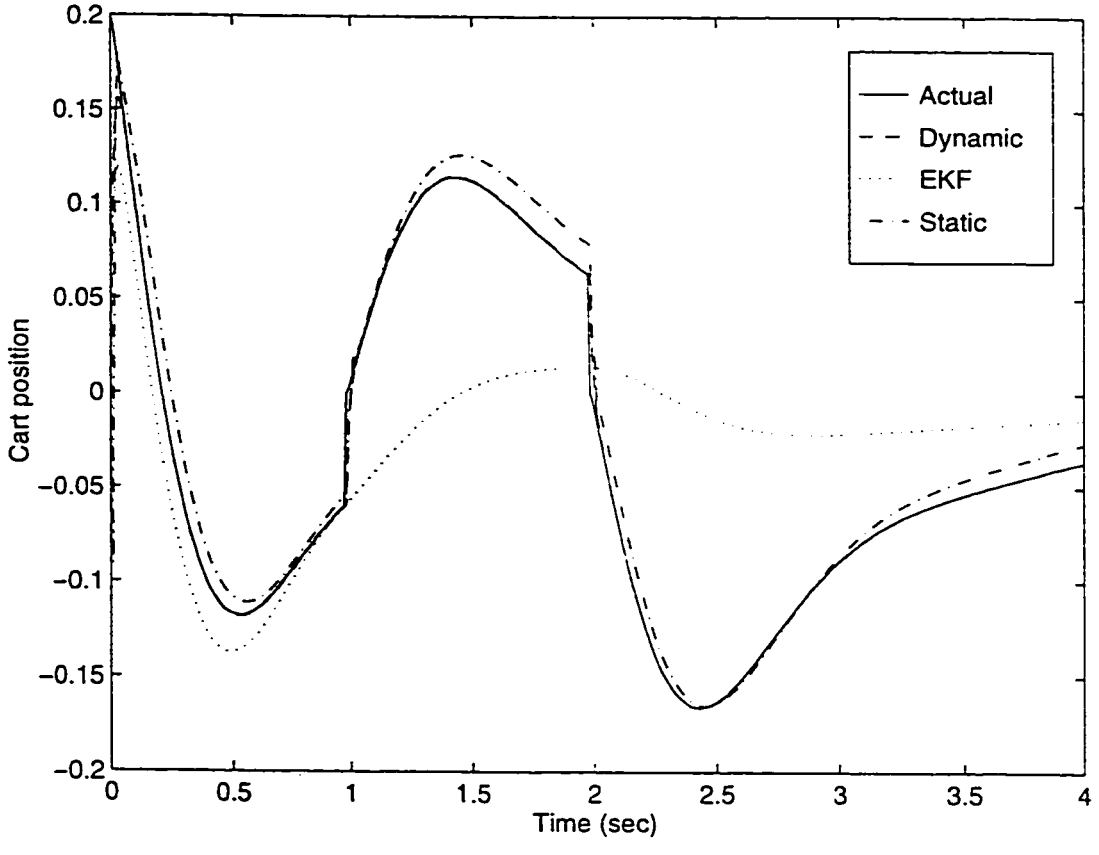


Figure 7.11: Estimated state 1 for the inverted pendulum problem

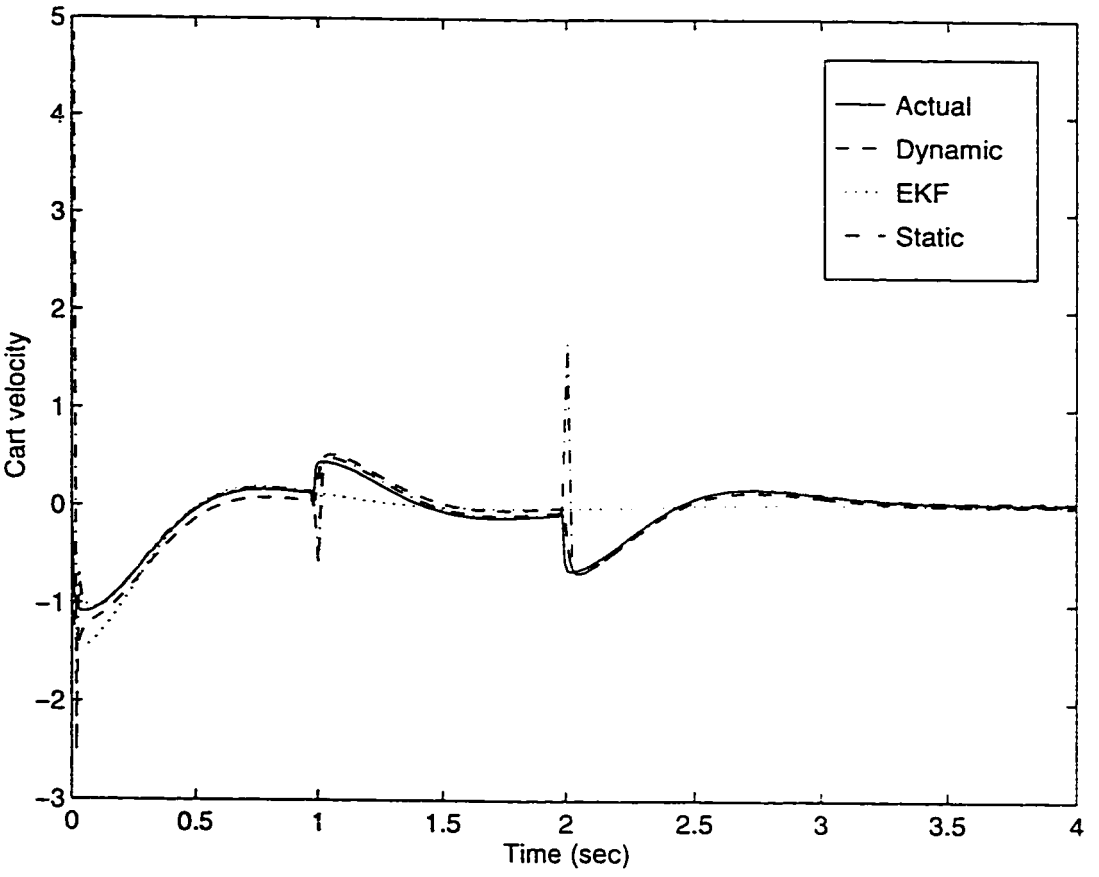


Figure 7.12: Estimated state 2 for the inverted pendulum problem

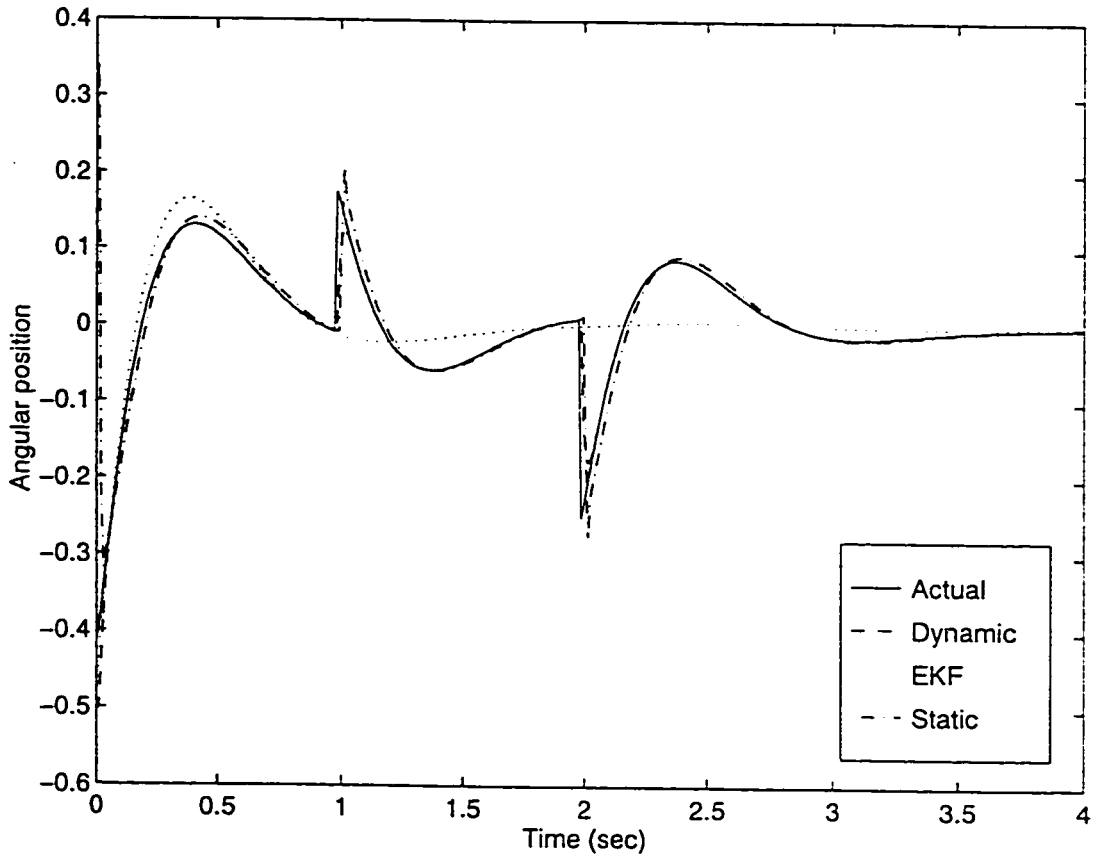


Figure 7.13: Estimated state 3 for the inverted pendulum problem

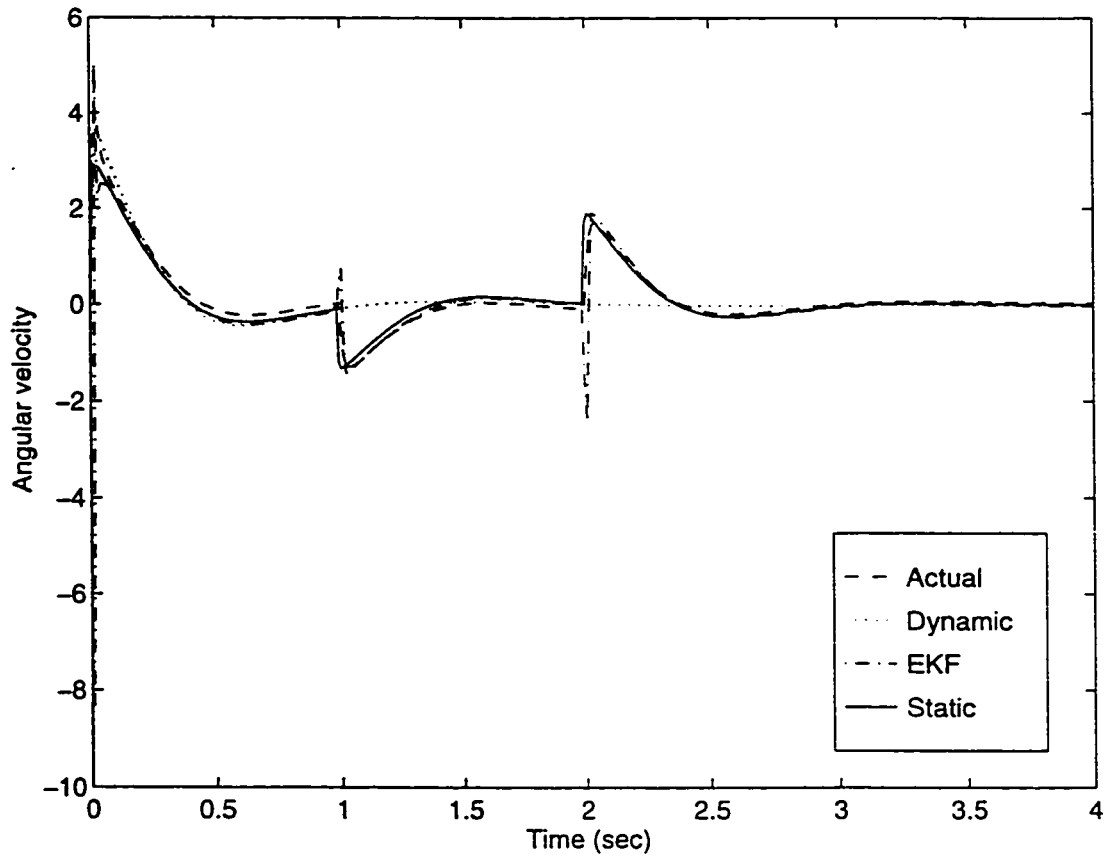


Figure 7.14: Estimated state 4 for the inverted pendulum problem

Table 7.19: SSE for the inverted pendulum without angular displacement

Σ_w	Σ_v	Dynamic	EKF	Static
0.0	0.0	9.3269	7.1549	222.1165
0.01	0.0	108.1171	53.9820	451.0704
0.0	0.01	10.0895	7.9582	394.3590
0.01	0.01	111.1129	54.9807	626.8287

Table 7.20: SSE for the perturbed inverted pendulum problem

	Dynamic	EKF	Static
$\Delta M = 0.045$	16.2415	85.9093	274.6809
$\Delta K_m = 0.01$	14.3175	63.9805	246.6937
$\Delta m = 0.1$	16.3672	85.7152	275.6959
$\Delta R_m = 0.5$	15.8926	86.0071	272.8404

angular variations of the inverted pendulum are considered during the simulation process, that is the inverted pendulum is allowed to run smoothly once it is initialized with a particular initial condition. The performance is shown in Figs 7.15 to 7.18 and for different noise levels the performance is given in Table 7.19.

7.3.2 Model perturbations

The model of the inverted pendulum is slightly varied and the effects on the observer schemes are observed based on the sum of the squared errors of the states. In all the cases we studied, the dynamic observer performed superior among all the schemes as can be verified by Table 7.20.

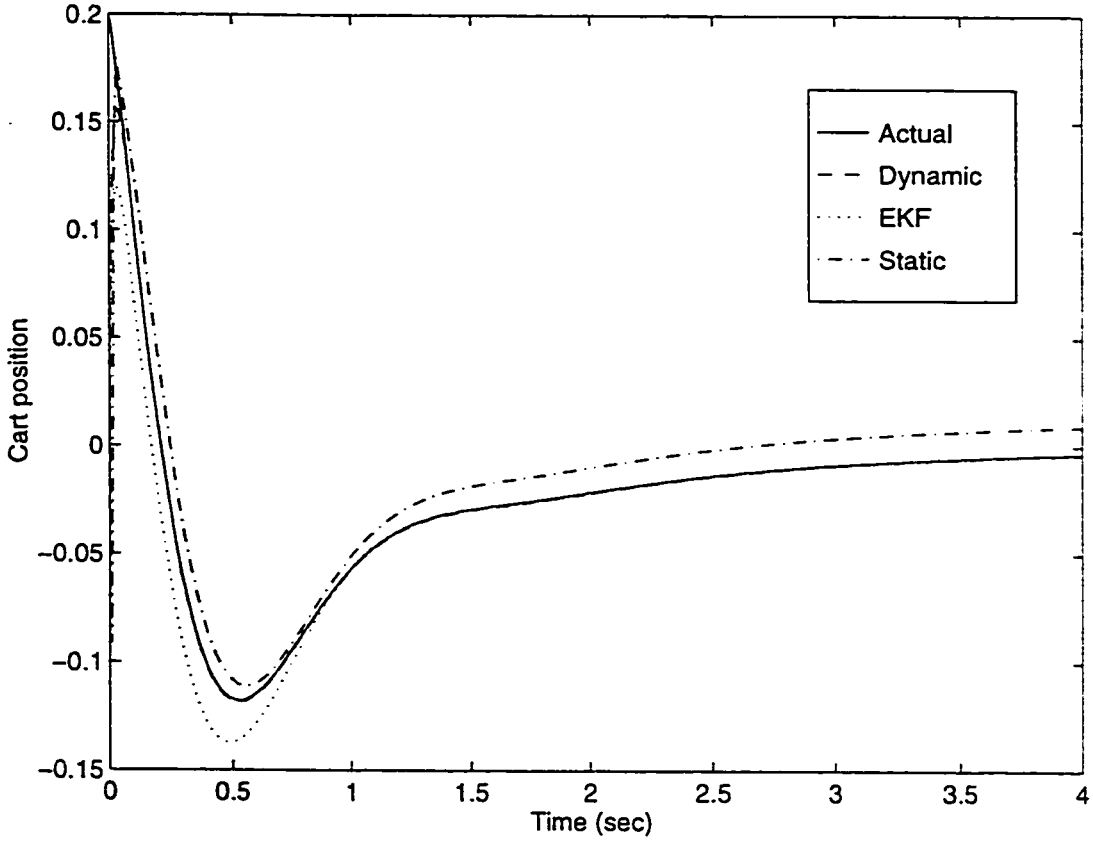


Figure 7.15: Estimated state 1 without angular disturbance

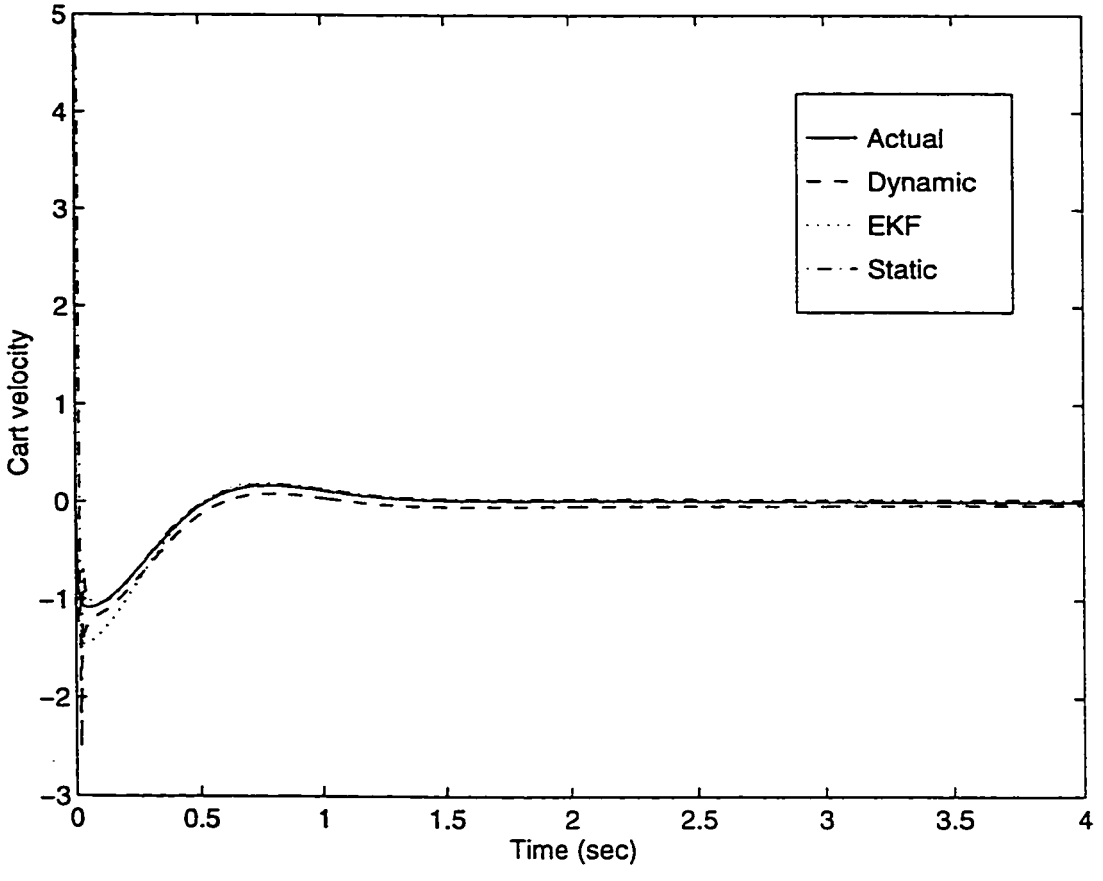


Figure 7.16: Estimated state 2 without angular disturbance

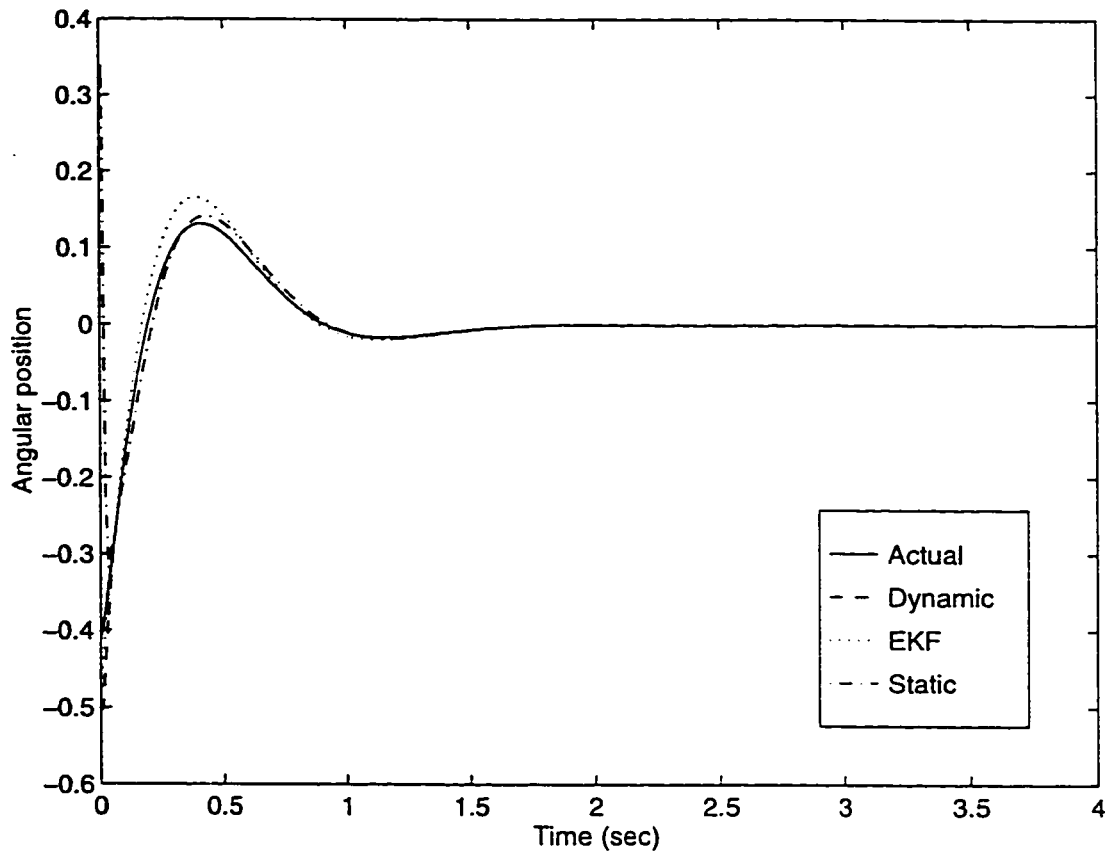


Figure 7.17: Estimated state 3 without angular disturbance

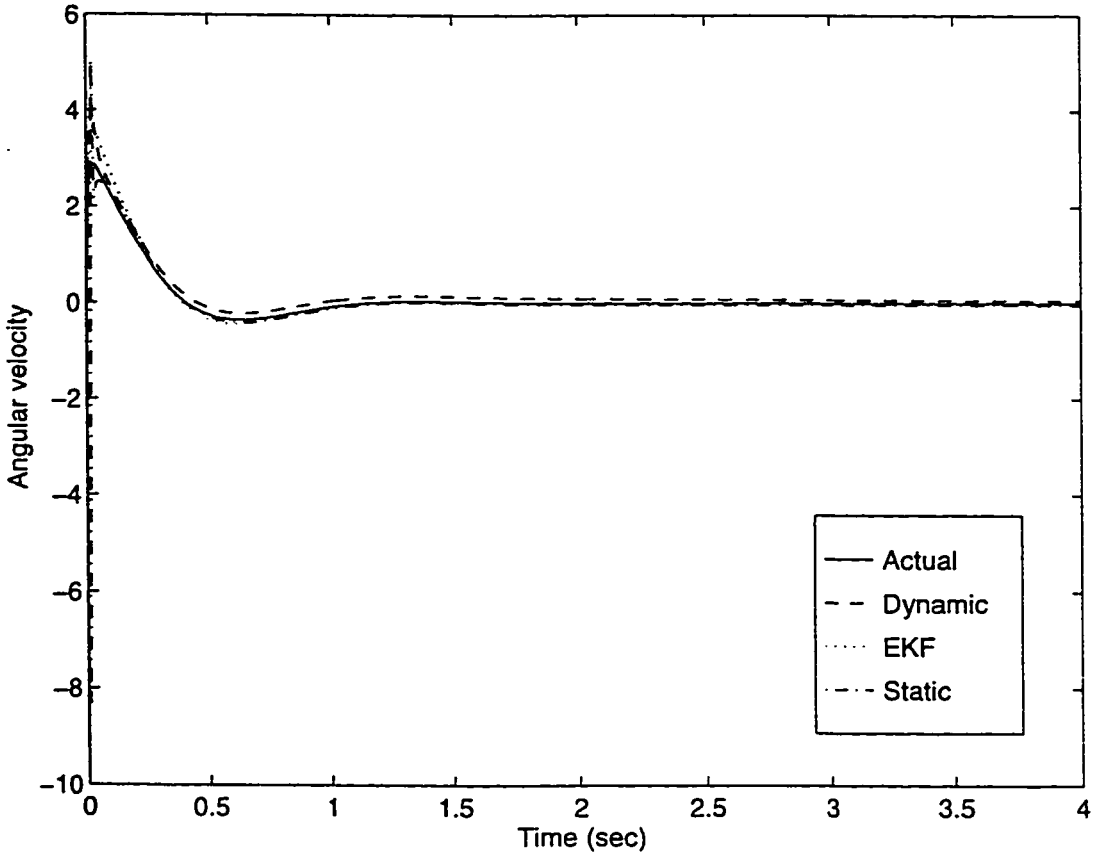


Figure 7.18: Estimated state 4 without angular disturbance

Chapter 8

Conclusion

The research work carried out in this thesis exploits the function approximation capabilities of neural networks in the area of nonlinear dynamical systems. Our concentration was in the design of dynamic neural observers for nonlinear systems.

8.1 Contributions

The main contributions of this work can be summarized as follows:

- Based on the linear observer design techniques, six different schemes for full and reduced order Dynamic Neural Observers for nonlinear plants are developed.
- The feasibility of the proposed schemes are evaluated by implementing the schemes on a number of systems and analyzing the performance of the pro-

posed observers.

- The Block Partial Derivatives are employed for the computation of the gradients during the training of the Neural observers to simplify the training procedure.
- Adaptive learning rate based on the linearization of the error in weight space, is used for improved convergence properties of the training algorithm.
- Comparative studies of all the schemes with each other, with static neural observer and Extended Kalman estimators are conducted.
- Robustness properties of the schemes are tested and compared in the presence of disturbances in the form of noise and parameter variations.

8.2 Conclusion

The schemes presented perform better than the static neural observer scheme. The variations in the six different proposed schemes are extensions of linear observer designs to nonlinear systems. EKF, performs better in the presence of noises but the performance deteriorates in the presence of modelling error. The design stage of the neural observers requires quite a lot of computations but during the implementation stage the calculations involved are very trivial. Whereas the Kalman filter is computationally relatively complex in the implementation stage. Commercially avail-

able neural chips can be used to reduce even further the calculations involved in the dynamic observers.

Some recommendations for the future work are:

- As an advancement in the proposed work, practical implementation can be undertaken to check the performance of the schemes in real time systems.
- Theoretical study can be performed to gain insight about local and global stability of the proposed schemes.
- The training procedure should be studied to reduce the time required for the training of the neural observers.

Bibliography

[1] Pieter Eykhoff. System identification. *John Wiley and Sons*, 1974.

[2] Narendra K.S. and Parthasarathy K. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4 27, 1990.

[3] L. Ljung and T. Soderstrom. Theory and practice of recursive identification. *The MIT press*, 1983.

[4] P.E.Mooral and J.W.Grizzle. Observer design for nonlinear systems with discrete time mearsurements. *IEEE transactions on automatic control*, 40(3):395 404, 1995.

[5] A. H. Jazwinski. Stochastic processes and filtering theory. *Academic Press*, 1970.

[6] M. S. Ahmed. An innovation representation for nonlinear systems with application to parameter and state estimation. *Automatica*, 30(12):1967 1974, 1994.

[7] Mandel. System identification. *Prentice Hall*, 1993.

[8] Boutayeb M., Rafaralahy H., and Darouach M. Convergence analysis of the extended kalman filter as an observer for nonlinear discrete time systems. *Proceedings of the 34th conference on decision and control*, pages 1555–1560, 1995.

[9] J. S. Dhingra, R. L. Moose, H. Vanlandingham, and T. A. Lauzon. A computationally efficient technique for state estimation of nonlinear systems. *Automatica*, 28(2):395–399, 1992.

[10] P.E.Mooral and J.W.Grazzle. Asymptotic observers for detectable and poorly observable systems. *Proceedings of the 34th conference on decision and control*, pages 108–114, 1995.

[11] Bestle D. and Zeitz M. Canonical form observer design for nonlinear time variable systems. *International Journal of control*, 38(2):419–431, 1983.

[12] Krener A. and Respondik W. Nonlinear observers with linearizable error dynamics. *SIAM journal of control and optimization*, 23(2), 1985.

[13] Ramnath R.V. and Paynter H.M. Scaling transformations in nonlinear systems. *Winter annual of the American society of mechanical engineers*, 1979.

[14] Nicosia S., Tomie P., and Tornambe. Feedback control of elastic robots by pseudolinearization techniques. *Proceedings of the 25th Conference on decision and control*. 1986.

[15] Reboulet C. and Champetier C. A new method for linearizing nonlinear systems: The pseudolinearization. *International Journal of control*, 40:631–638, 1984.

[16] Bauman J. and Rugh W. Feedback control of nonlinear systems by extended linearization. *IEEE transactions on automatic control*. AC31(1), 1986.

[17] Misawa E. and Hedrick J. Nonlinear observers - a state of the art survey. *Transactions of the ASME*, 111:344–352, 1989.

[18] Thau F. E. Observing the state of nonlinear dynamical systems. *International Journal of control*, 17(3), 1973.

[19] S. H. Crandall. On statistical linearization for nonlinear oscillators. *Winter annual meeting of the American Society of Mechanical Engineers*, 1979.

[20] Walcott B., Corless M., and Zak S. Comparative study of nonlinear state observation techniques. *International Journal of control*, 45(6):2109-2132, 1987.

[21] Zeitz M. The extended luenberger observer for nonlinear systems. *Systems and control Letters*, 9:149–156.

[22] Kazantzis N. and Kravaris C. A nonlinear luenberger-type observer with application to catalyst activitty estimation. *Proceedings of the American control Conference*, pages 1756–1761, 1995.

[23] Ciccarella G., Dalla Mora M., and Germani A. A luenberger-like observer for nonlinear systems. *International Journal of control*, 57(3):537–556, 1993.

[24] Gauthier J.P., Hammouri H., and Othman S. A simple observer for nonlinear systems applications to bioreactors. *IEEE transactions on automatic control*. 37(6):875–880, 1992.

[25] Ciccarella G., Dalla Mora M., and Germani A. Observers for discrete-time nonlinear systems. *Systems and control letters*, 20:373–382, 1993.

[26] Drakunov S.V. An adaptive quasioptimal filter with discontinuous parameteres. *Automation and remote control*. 44(2):76–86, 1983.

[27] Slotine J.J., Hedrik J.K., and Misawa E.A. On sliding observers. *Proceedings of the American control conference*, 1986.

[28] Slotine J.J., Hedrik J.K., and Misawa E.A. Nonlinear state estimation using sliding observers. *Proceedings of the 25th conference on Decision and control*. 1986.

[29] Drakunov S.V. and Utkin V. Sliding mode observers. tutorial. *Proceedings of the 34th conference on Decision and control*. pages 3376 3378. 1995.

[30] Aitkin V.C. and Schwartz H.M. Towards robust discrete time sliding mode observers. *Proceedings of the American control conference*. pages 3730-3734, 1995.

[31] Porter L. and Passino K. Genetic adaptive observers. *Proceedings of the American control conference*, pages 1847-1851. 1995.

[32] Page G.F., Gomm J.B., and Williams D. Application of neural networks to modelling and control. *Chapman and Hall.London*. 1993.

[33] Cybenko G. Approximation by superposition of sigmoidal function. *Mathematics of control signals and systems*, 2(4):303-314. 1989.

[34] Hornik K., Stinchcomb M., and White H. Multilayer feedforward networks are universal approximators. *Neural networks*. 2:359 366. 1989.

[35] Hunt K.J., Sbarbaro D., Zbikowski R., and Gawthrop P.J. Neural networks for control systems - a survey. *Automatica*. 28(6):1003-1112. 1992.

[36] S. Chen, S. A. Billing, and P. M. Grant. Nonlinear system identification using neural networks. *International journal of control*. 51(6):1191 1214. 1990.

[37] Werbos P.J. Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10):1550-1560. 1990.

[38] Narendra K.S. and Parthasarathy K. Gradient method for the optimization of dynamical systems containing neural networks. *IEEE Transactions on Neural Networks*, 2:252-262, 1991.

[39] Willis M.J., Montague G.A., Massimo D., Tham M.T., and Morris A.J. Artificial neural networks in process estimation and control. *Automatica*, 28(6):1181-1187, 1992.

[40] Asriel U. Levin and Kumpati S. Narendra. Recursive identification using feedforward neural networks. *International journal of control*, 61(3):533-547, 1995.

[41] Levin A.U. and Narendra K.S. Control of nonlinear dynamical systems using neural networks: Controllability and stabilization. *IEEE Transactions on Neural Networks*, 4(2):192-206, 1993.

[42] Levin A.U. and Narendra K.S. Control of nonlinear dynamical systems using neural networks - part ii: Observability, identification, and control. *IEEE Transactions on Neural Networks*, 7(1):30-42, 1996.

[43] M. S. Ahmed. Block partial derivative and its application to neural-net-based direct-model-reference adaptive control. *IEE proceedings - Control theory Appl.*, 141(5):305–314, 1994.

[44] M. S. Ahmed. Bpd computation and model reference adaptive control (mrac) of hammerstein plant. *IEE proceedings - Control theory Appl.*, 142(5):475–485, 1995.

[45] Rumelhart D. E., G. E. Hinton, and R. J. Williams. Learning internal representation by error propagation. *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, 1:318–362, 1986.

[46] Su H. Qin S. and McAvoy T.J. Comparison of four neural net learning methods for dynamic system identification. *IEEE Transactions on Neural Networks*, 3(1):122–130, 1992.

[47] L. Fausett. Fundamentals of neural networks. *Prerntice Hall*, 1994.

[48] Simon Haykin. Neural networks a comprehensive foundation. *Macmillan Publishing Company*, 1994.

[49] A.V. Oppenheim and R.W. Schafer. Discrete-time signal processing. *Prentice-Hall*, 1989.

[50] Chi-Tsong Chen. Linear system theory and design. *Saunders College Publishing*. 1984.

[51] Karl J Astrom and Bjorn Wittenmark. Computer controlled systems theory and design. *Prentice Hall Inc.*, 1990.

[52] Phillips C.L. and Nagle H.T. Digital control system analysis and design. *Prentice Hall Inc.*, 1990.

[53] M.S. Ahmed and I. A. Tasadduq. Neural net controller for nonlinear plants a design approach through linearization. *IEE proceedings for control theory and applications*, 141(5):315–322, 1994.

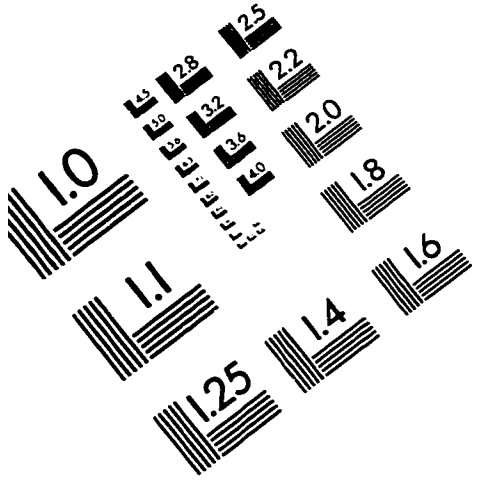
[54] Ahmed M.S. Neural controller design for nonlinear plants. *Working paper*, 1997.

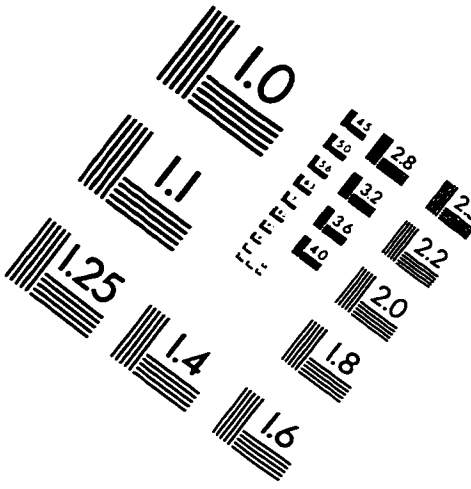
[55] M.S. Ahmed and M. Anjum. Learning rate algorithm for neural net based direct adaptive control. *Arabian journal for science and engineering*, 18(4):493 513, 1993.

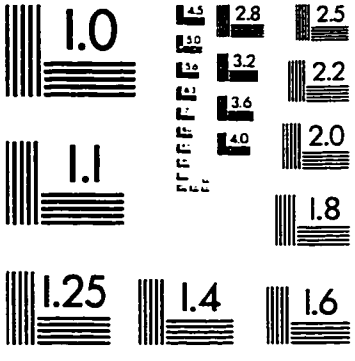
Vita

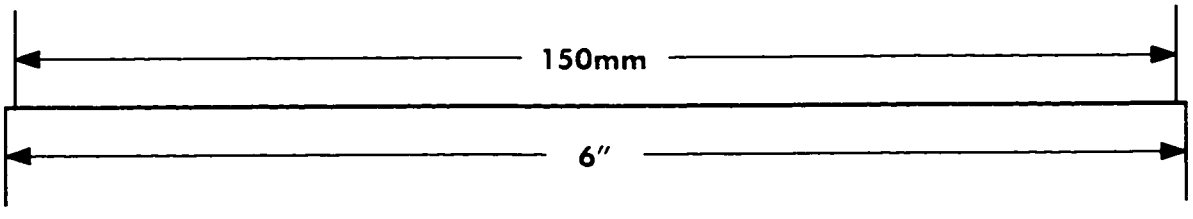
- Sayyid Hasan Riyaz
- Born in Karachi, Pakistan.
- Received Bachelor's degree in Electrical Engineering from the N. E. D. University of Engineering and Technology, Karachi, Pakistan in March, 1991.
- Worked in Zelin Pakistan Ltd. Karachi, for one year. Later on joined Ministry of Electricity and Water, Kuwait, for two years.
- Completed Master's degree requirements at King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia in November, 1997.

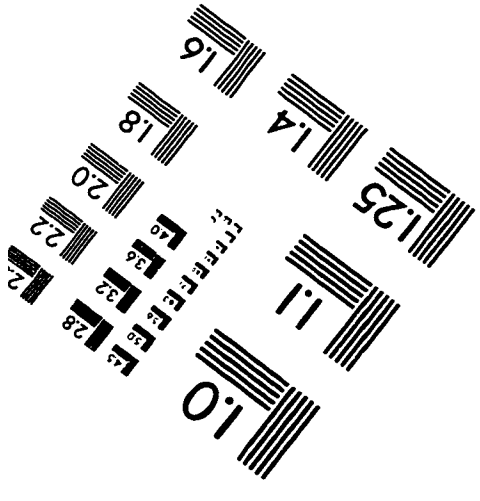
IMAGE EVALUATION TEST TARGET (QA-3)

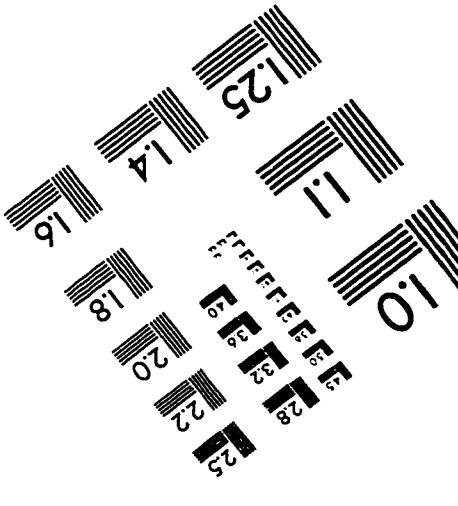












APPLIED IMAGE, Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989